ESCUELA POLITÉCTICA SUPERIOR

DPTO. DE INFORMÁTICA

DOCTORADO EN INGENIERÍA INFORMÁTICA Y TELECOMUNICACIÓN

THESIS

A NEW ANT COLONY OPTIMIZATION MODEL FOR COMPLEX GRAPH-BASED PROBLEMS

Author

ANTONIO GONZÁLEZ PARDO

Advisor

Dr. D. David Camacho Fernández

July 2014

UNIVERSIDAD AUTONOMA
DE MADRID

Ciudad Universitaria de Cantoblanco
Escuela Politécnica Superior
Dpto. de Ingeniería Informática
C/Francisco Tomás y Valiente, 11
28049 Madrid (Spain) Telf: +34 91 497 24 19
Fax: +34 91 497 22 35

Dr. D. David Camacho Fernandez, Profesor Titular de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid,

CERTIFICA

Que la tesis "*A New Ant Colony Optimization Model for Complex Graph-based Problems*", presentada por D. Antonio González Pardo y realizada en el Departamento de Ingeniería Informática, reúne méritos suficientes para optar al grado de Doctor, por lo que puede procederse a su depósito y lectura.

Madrid, Julio 2014.

Fdo.: Dr. D. David Camacho Fernández

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Autónoma de Madrid, el día ..... de ..... de 2014.

Presidente	D. .....
Vocal	D. .....
Vocal	D. .....
Vocal	D. .....
Secretario	D. .....

Realizado el acto de defensa y lectura de la Tesis el día ..... de ..... de 2014 en .....

Calificación: .....

EL PRESIDENTE EL SECRETARIO

LOS VOCALES

A mis padres y a Coraima,
por su apoyo, cariño y dedicación
en todos estos años.
Os quiero.

El primer paso para conseguir algo es creer en ello.

Abstract

Nowadays, there is a huge number of problems that due to their complexity have employed heuristic-based algorithms to search for near-to-optimal (or even optimal) solutions. These problems are usually *NP-complete*, so classical algorithms are not the best candidates to address these problems because they need a large amount of computational resources, or they simply cannot find any solution when the problem grows. Some classical examples of these kind of problems are the *Travelling Salesman Problem* (TSP) or the *N-Queens* problem. It is also possible to find examples in real and industrial domains related to the optimization of complex problems, like *planning*, *scheduling*, *Vehicle Routing Problems* (VRP), *WiFi network Design Problem* (WiFiDP) or behavioural pattern identification, among others.

Regarding to heuristic-based algorithms, two well-known paradigms are *Swarm Intelligence* and *Evolutionary Computation*. Both paradigms belongs to a subfield from *Artificial Intelligence*, named *Computational Intelligence* that also contains *Fuzzy Systems*, *Artificial Neural Networks* and *Artificial Immune Systems* areas.

Swarm Intelligence (SI) algorithms are focused on the collective behaviour of self-organizing systems. These algorithms are characterized by the generation of collective intelligence from non-complex individual behaviour and the communication schemes amongst them. Some examples of SI algorithms are *particle swarm optimization*, *ant colony optimization* (ACO), *bee colony optimization* o *bird flocking*.

Ant Colony Optimization (ACO) are based on the foraging behaviour of these insects. In these kind of algorithms, the ants take different decisions during their execution that allows them to build their own solution to the problem. Once any ant has finished its execution, the ant goes back through the followed path and it deposits, in the environment, *pheromones* that contains information about the built solution. These pheromones will influence the decision of future ants, so there is an indirect communication through the environment called *stigmergy*.

When an ACO algorithm is applied to any of the optimization problems just described, the problem is usually modelled into a graph. Nevertheless, the classical graph-based representation is not the best one for the execution of ACO algorithms because it presents some important pitfalls. The first one is related to the polynomial, or even exponential, growth of the resulting graph. The second pitfall is related to those problems that needs from real variables because these problems cannot be modelled using the classical graph-based representation.

On the other hand, *Evolutionary Computation* (EC) are a set of population-based algorithms based in the Darwinian evolutionary process. In this kind of algorithms there is one (or more) population composed by different individuals that represent a possible solution to the problem. For each iteration, the population evolves by the use of evolutionary procedures which means that better individuals (i.e. better solutions) are generated along the execution of the algorithm. Both kind of algorithms, EC and SI, have been traditionally applied in previous NP-hard problems. Different population-based strategies have been developed, compared and even combined to design hybrid algorithms.

This thesis has been focused on the analysis of classical graph-based representations and its application in ACO algorithms into complex problems, and the development of a new ACO model that tries to take a step forward in this kind of algorithms. In this new model, the problem is represented using a reduced graph that affects to the ants behaviour, which becomes more complex. Also, this size reduction generates a fast growth in the number of pheromones created. For this reason, a new metaheuristic (called *Oblivion Rate*) has been designed to control the number of pheromones stored in the graph.

In this thesis different metaheuristics have been designed for the proposed system and their performance have been compared. One of these metaheuristics is the Oblivion Rate, based on an exponential function that takes into account the number of pheromones created in the system. Other Oblivion Rate function is based on a bio-inspired swarm algorithm that uses some concepts extracted from the evolutionary algorithms. This bio-inspired swarm algorithm is called *Coral Reef Opmization* (CRO) algorithm and it is based on the behaviour of the corals in a reef.

Finally, to test and validate the proposed model, different domains have been used such as the *N-Queens Problem*, the *Resource-Constraint Project Scheduling Problem*, the Path Finding problem in Video Games, or the Behavioural Pattern Identification in users. In some of these domains, the performance of the proposed model has been compared against a classical *Genetic Algorithm* to provide a comparative study and perform an analytical comparison between both approaches.

Resumen

En la actualidad, existen un gran número de problemas que debido a su complejidad necesitan algoritmos basados en heurísticas para la búsqueda de solucionas subóptimas (o incluso óptimas). Normalmente, estos problemas presentan una complejidad *NP-completa*, por lo que los algoritmos clásicos de búsqueda de soluciones no son apropiados ya que necesitan una gran cantidad de recursos computacionales, o simplemente, no son capaces de encontrar alguna solución cuando el problema crece. Ejemplos clásicos de este tipo de problemas son el problema del vendedor viajero (o TSP del inglés *Travelling Salesman Problem*) o el problema de las N-reinas. También se pueden encontrar ejemplos en dominios reales o industriales que generalmente están ligados a temas de optimización de sistemas complejos, como pueden ser problemas de planificación, *scheduling*, problemas de enrutamiento de vehículos (o VRP del inglés *Vehicle Routing Problem*), el diseño de redes Wifi abiertas (o WiFiDP del inglés *WiFi network Design Problem*), o la identificación de patrones de comportamiento, entre otros.

En lo referente a los algoritmos basados en heuristicas, dos paradigmas muy conocidos son los algoritmos de enjambre (*Swarm Intelligence*) y la computación evolutiva (*Evolutionary Computation*). Ambos paradigmas pertencen al subárea de la Inteligencia Artificial denominada *Inteligencia Computacional*, que además contiene los sistemas difusos, redes neuronales y sistemas inmunológicos artificiales.

Los algoritmos de inteligencia de enjambre, o *Swarm Intelligence*, se centran en el comportamiento colectivo de sistemas auto-organizativos. Estos algoritmos se caracterizan por la generación de inteligencia colectiva a partir del comportamiento, no muy complejo, de los individuos y los esquemas de comunicación entre ellos. Algunos ejemplos son *particle swarm optimization*, *ant colony optimization* (ACO), *bee colony optimization* o *bird flocking*.

Los algoritmos de colonias de hormigas (o ACO del inglés *Ant Colony Optimization*) se basan en el comportamiento de estos insectos en el proceso de recolección de comida. En este tipo de algoritmos, las hormigas van tomando decisiones a lo largo de la simulación que les permiten construir su propia solución al problema. Una vez que una hormiga termina su ejecución, deshace el camino andado depositando en el entorno *feronomas* que contienen información sobre la solución construida. Estas feromonas influirán en las decisiones de futuras hormigas, por lo que produce una comunicación indirecta utilizando el entorno. A este proceso se le llama *estigmergia*.

Cuando un algoritmo de hormigas se aplica a alguno de los problemas de optimización

descritos anteriormente, se suele modelar el problema como un grafo sobre el cual se ejecutarán las hormigas. Sin embargo, la representación basada en grafos clásica no parece ser la mejor para la ejecución de algoritmos de hormigas porque presenta algunos problemas importantes. El primer problema está relacionado con el crecimiento polinómico, o incluso expnomencial, del grafo resultante. El segundo problema tiene que ver con los problemas que necesitan de variables reales, o de coma flotante, porque estos problemas, con la representación tradicional basada en grafos, no pueden ser modelados.

Por otro lado, los algoritmos evolutivos (o EC del inglés *Evolutionary Computation*) son un tipo de algoritmos basados en población que están inspirados en el proceso evolutivo propuesto por Darwin. En este tipo de algoritmos, hay una, o varias, poblaciones compuestas por individuos diferentes que representan problems solutiones al problema modelado. Por cada iteración, la población evoluciona mediante el uso de procedimientos evolutivos, lo que significa que mejores individuos (mejores soluciones) son creados a lo largo de la ejecución del algoritmo. Ambos tipos de algorithmos, EC y SI, han sido tradicionalmente aplicados a los problemas *NP-completos* descritos anteriormente. Diferentes estrategias basadas en población han sido desarrolladas, comparadas e incluso combinadas para el diseño de algoritmos híbridos.

Esta tesis se ha centrado en el análisis de los modelos clásicos de representación basada en grafos de problemas complejos para la posterior ejecución de algoritmos de colonias de hormigas y el desarrollo de un nuevo modelo de hormigas que pretende suponer un avance en este tipo de algoritmos. En este nuevo modelo, los problemas son representados en un grafo más compacto que afecta al comportamiento de las hormigas, el cual se vuelve más complejo. Además, esta reducción en el tamaño del grafo genera un rápido crecimiento en el número de feronomas creadas. Por esta razón, una nueva metaheurística (llamada *Oblivion Rate*) ha sido diseñada para controlar el número de feromonas almacenadas en el grafo.

En esta tesis, varias metaheuristicas han sido diseñadas para el sistema propuesto y sus rendimientos han sido comparados. Una de estas metaheurísticas es la *Oblivion Rate* basada en una función exponencial que tiene en cuenta el número de feromonas creadas en el sistema. Otra *Oblivion Rate* está basada en un algoritmo de enjambre bio-inspirado que usa algunos conceptos extraídos de la computación evolutiva. Este algoritmo de enjambre bio-inspirado se llama Optimización de arrecifes de corales (o CRO del inglés *Coral Reef Optimization*) y está basado en el comportamiento de los corales en el arrecife.

Finalmente, para validar y testear el modelo propuesto, se han utilizado diversos dominios de aplicación como son el problema de las N-reinas, problemas de planificación de proyectos con restricciones de recursos, problemas de búsqueda de caminos en entornos de videojuegos y la identificación de patrones de comportamiento de usuarios. En algunos de estos dominios, el rendimiento del modelo propuesto ha sido comparado contra un algoritmo genético clásico para realizar un estudio comparativo, y analítico, entre ambos enfoques.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

This chapter provides a general overview of this PhD dissertation. Section 1.1 motivates the PhD work carried out. Section 1.2 describes the problem that has motivated this work and finally, Section 1.3 describes the different objectives in which this dissertation is focused.

1.1 Motivation

Nowadays, there exist a wide number of complex problems of special interest in the research community and the industry. These problems can be roughly classified in fundamental (or classical) research and industrial problems. Some examples of classical research problems are *Travelling Salesman Problem* (TSP) [Dorigo and Gambardella, 1997a, Dorigo and Gambardella, 1997b], graph coloring problem [Costa and Hertz, 1997, Dowsland and Thompson, 2008], knapsack problems [Alaya et al., 2004, Ke et al., 2010] or subgraph isomorphism problem [Zampelli et al., 2010, Sorlin and Solnon, 2004], among others. In industrial domains the problems are mainly focused on the optimization of specific functions or procedures. Examples of these kind of problems are, Project Scheduling Problems (PSP) [Merkle et al., 2002, Xing et al., 2010], Car Sequencing [Gottlieb et al., 2003, Morin et al., 2009], Vehicle Routing Problems (VRP) [Bell and McMullen, 2004] or Video Games [Fernandes et al., 2008, Gonzalez-Pardo et al., 2014a].

The main common characteristic of previous problems is that all of them are *NP-complete* problems. This means that the time required to solve the problem using any algorithm increases very quickly as the size of the problem grows (usually the time to solve the problem exponentially increases whereas the size of the problem linearly grows).

NP stands for Non-deterministic Polynomial time. This means that the problem can be solved in Polynomial time using a Non-deterministic Turing machine. Basically, a solution has to be testable in polynomial time. The main problem with NP-Complete (or NP-hard) problems is that are not solvable in realistic time. For this reason, it is really common the use of heuristic algorithms that explore only part of the solution space to find solutions faster than force-brute algorithms. Some examples of heuristic algorithms are the ones belonging to *Swarm Intelligence* and *Evolutionary Computation* research field, both areas are usually included as part of *Computational Intelligence*.

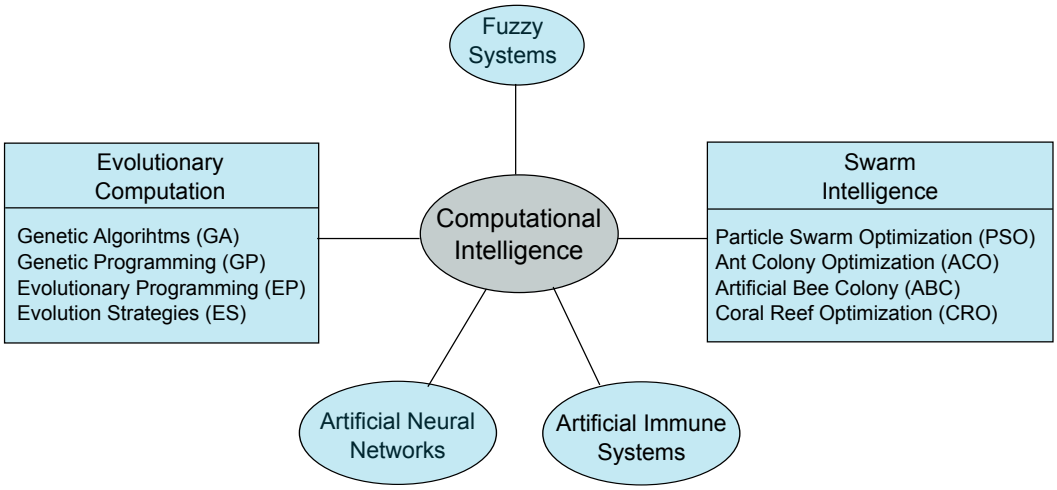


Figure 1.1: General classification of different research areas that belongs to Computational Intelligence field.

Computational Intelligence(CI) [Engelbrecht, 2007] can be defined as a set of bio-inspired research areas focused on the study of adaptive mechanisms to enable, or facilitate, intelligent behaviour in complex and changing environments [Engelbrecht, 2007]. Figure 1.1 shows the different research fields that compose CI, which are Fuzzy Systems [Kosko, 1992], Evolutionary Computation [Bäck et al., 1997], Artificial Neural Networks [Hornik et al., 1989], Artificial Immune Systems [Forrest et al., 1994] and Swarm Intelligence [Bonabeau et al., 1999]. This thesis is focused on *Ant Colony Optimization* (ACO) algorithms that belongs to *Swarm Intelligence* research field, and its comparison against other population-based algorithms from *Evolutionary Computation*, in this case, *Genetic Algorithms* and *Coral Reef Optimization* algorithm (CRO).

Evolutionary Computation (EC) takes the darwinian idea of natural selection as a basis to design algorithms, generally known as *Evolutionary Algorithms* (EAs). In these algorithms there is a population of individuals (potential solutions) which generates new individuals by the reproduction procedures (i.e. crossover and mutation operators). Probably, the most popular EC algorithm is Genetic Algorithms (GAs) [Goldberg, 1989, Holland, 1992].

EA algorithms use a *population* of individuals that are evolved during the execution. Each individual represents a possible solution to the problem and it is composed by the *chromosome* that defines the characteristic of the individual. For each generation, the better individuals produce the offspring that will compose the next population. New individuals are generated taking into account the characteristic (genotype) of their parents.

The generation of a new offspring is performed by two reproduction procedures. The first one, called *crossover*, generates a new individual by combining parts of the parents genotypes. Once this procedure has finished, the procedure called *mutation* changes randomly some parts in the offspring genotype.

There are different families of EA algorithms depending on how the information contained in the genotype is represented. Also, depending on how the information is encoded into the genotype, the reproduction mechanisms may change. The four main families of EA algorithms are the following:

- **Genetic Algorithms** (GA). In this type of algorithms, the solution is encoded in a linear vector (usually binary or integer) [Holland, 1992, Goldberg, 1989].
- **Genetic Programming** (GP). Here the population contains a collection of programs represented by trees [Koza, 1992, Poli et al., 2008].
- **Evolutionary Programming** (EP), where the structure that is evolved is based on a finite automata [Fogel, 1962, Fogel and Fogel, 1996].
- **Evolutionary Strategies** (ES) are focused on the numeral optimization in the space of real numbers. [Rechenberg, 1965, Beyer and Schwefel, 2002]

On the other hand, *Swarm Intelligence* (SI) is originated from the study of colonies, or swarms, of social organisms. Individuals of the swarm are relatively simple, but their collective behaviour is usually very complex. This complex behaviour is generated by pattern interactions between the individuals and it is not easily to predict. This characteristic is called *emergence*, and it can be defined as the process of generating some knowledge (for example, structures, patterns or properties) in complex systems without any coordinated control system.

The collective behaviour of a swarm emerges from the behaviour of the individuals. This means that there is an influence between the behaviour of the individual and the behaviour of the swarm. Individual decisions affects to the behaviour of the swarm, but, at the same time, the behaviour of the swarm affects to the behaviour of the individuals.

The communication between individuals can be performed in two different ways. The communication can be direct (by physical contact or visual inputs) or indirect by changing the environment where the individuals are executed. The term *stigmergy* is used to defined the indirect communication between individuals.

The most commonly known family algorithms that composed SI are:

- **Ant Colony Optimization** (ACO) algorithms are based on the behaviour of ants seeking a path from the nest to the source of food [Dorigo et al., 1996, Dorigo and Stützle, 2001].
- **Particle Swarm Optimization** (PSO) is a search algorithm based on the simulation of the social behaviour of birds within a flock [Kennedy and Eberhart, 1995, Poli et al., 2007].
- **Artificial Bee Colony** (ABC) is focused on the intelligent foraging behaviour of bee swarms [Teodorović and Dell'Orco, 2005, Karaboga and Basturk, 2007]

There are, also, some hybrid approaches that are based on some concepts taken from the *Evolutionary Computation*, and other concepts from *Swarm Intelligence*. One of these approaches is called **Coral Reef Optimization** (CRO) algorithm. Although, in Figure 1.1, CRO algorithm is considered an approach belonging to the Swarm Intelligence research field, it has some characteristics that makes CRO algorithm an evolutionary-based algorithm. CRO is a bio-inspired algorithm based on the behaviour of the corals in a reef. This novel search and optimization algorithm is composed by a set of individuals, called *corals*, that reproduce and the new corals "fight" to find a place in the reef [Salcedo-Sanz et al., 2013c, Salcedo-Sanz et al., 2013b].

1.2 Graph-Based Modelling in ACO

Some of the problems cited in previous section (Section 1.1), like graph coloring problem, knapsack problems, Project Scheduling Problems (PSP) or Video Games, can be modelled using a graph-based approach wherein different heuristic algorithms can be executed, such as ACO algorithms.

For this particular case, when the ants are travelling through the graph, they are building a solution to the problem. Nevertheless, the main problem regarding to this procedure is that depending on the problem, the resulting graph could be unmanageable, one of these traditional problems is the *Constraint Satisfaction Problems* (CSP).

Some of the application domains used in the experimental phase of this thesis, can be modelled as a CPS-based problem. In these problems, there is a set of variables that must be assigned some values, and a set of constraint that must be satisfied. Traditionally, the way that any CSP problem is modelled into a graph is creating a full-connected graph where there are as many nodes as pairs $\langle variable, value \rangle$ are contained in the problem. Using this representation two problems can be identified. The first one is regarding the size of the resulting graph, that it could be unmanageable. The second problem is that continuous problems, or problems where continuous variables are involved, cannot be represented.

Some other application domains, taken into account in this thesis like video games, are not CSP-based problems but they are also modelled into a graph for the execution of ACO algorithms. In this case, although the approach for modelling the problem as a graph is completely different as the approach just described, the size of the resulting graph could be, also, unmanageable.

This thesis has been focused on the analysis of the current graph-based models for ACO algorithms, and a new graph-based model for the execution of ACO algorithm is proposed. This new model, whose goal is to alleviate the mentioned problems, is composed by the representation of the problem as a graph, a new behaviour for the ants, and a new metaheuristic whose goal is to control the fast growing of the number of pheromones deposited in the graph. Finally, the model is validated in different domains.

1.3 Objective of the thesis

It can be summarized that there is a wide range of real complex problems that needs from heuristic algorithms to be solved. One of these heuristic algorithms is the ACO-based methods that are based on the behaviour of ants looking for food sources.

In order to execute ACO algorithms, the problem must be defined as a graph, but the way that the problem is modelled into a graph presents some pitfalls that need to be studied and analysed.

The main goal of this thesis can be summarized in the study and analysis of current models and heuristics in ACO algorithms and their utilization into real complex graph-based domains.

This general goal can be decomposed in the following subgoals:

S1 To review the state of the art related to Computational Intelligence methods, specially focused on Ant Colonies and Evolutionary Computation algorithms. Their representations, their main working features, and their application in basic research and industrial domains.

S2 To identify the strengths and weaknesses of the corresponding (graph-based) representations.

S3 To design a new graph-based representation that could improve, or alleviate, the identified weaknesses.

S4 To analyse and design new ants behaviours.

S5 To study and design new metaheuristics to improve the pheromone growth in the new proposed ACO model.

S6 To study and design new metaheuristics for ACO algorithms inspired by GA.

S7 To define a new ACO model based on a new graph-based representation, some new ants behaviour and new metaheuristics that will be used to control the pheromone growth in the graph.

S8 To validate the new proposed model in classical problems.

S9 To study and analyse the behaviour of previous graph-based model and pheromone control against other related algorithms, such as GA and CRO, when they are applied in complex problems.

S10 To empirically and experimentally compare the new ACO model proposed against GA.

1.4 Structure of the thesis

This thesis has been divided into eight different chapters, a short summary for each of them is provided below:

- This chapter (Chapter 1) is dedicated to introduce the main research questions and the thesis structure. In this chapter the motivation of this thesis, the main problem regarding to the graph-based modelling of complex problems for the execution of ACO algorithms and the different objectives of this thesis has been presented.
- Chapter 2 provides a conceptual framework about the State of the Art in the different research fields, and the application domains that have been used in this thesis. The different research fields explained in this chapter are; *Ant Colony Optimization*; *Evolutionary Computation*; and *Coral Reef Optimization* algorithm. The application domains can be grouped in three main categories: *CSP-based* problems (like *N-Queens* problem, and *Resource-Constraint Satisfaction Problem*), Video Games and Educational serious games.
- In Chapter 3, a detailed description about how complex problems (like the ones described in Chapter 2) are modelled into a restricted graph. This section also describes the *Constraint-Satisfaction Problems*, and how researchers have deal with CSP problems using *Evolutionary Computation* and *Ant Colony Optimization* algorithms.
- In this thesis a new ACO model for complex graph-based problems has been proposed. A detailed description related to this new model is given in Chapter 4. This chapter describes how the decision graph for executing the ACO algorithm is built, what is the new ant behaviours, and the different Oblivion Rate functions used in this thesis.
- Chapter 5 provides the experimental results of the proposed model for the *N-Queens* problems, the *Resource-Constraint Project Scheduling Problem*, the *Lemmings* game and the avatar behaviours in Virtual Worlds.
- The main conclusions obtained from this thesis and the future lines of work are described in Chapter 6 and Chapter 7 (in Spanish).
- Finally, Chapter 8 shows a short summary related to the different international publications derived from this thesis work and where these contributions can be found in the thesis.

CHAPTER 2

STATE OF THE ART

This chapter is devoted to contextualize the current state of the art in the basic research areas that constitute the core of this thesis, providing a general perspective on the key concepts, current trends and some limitations in those areas. In section 2.1 provides a review of the bio-inspired algorithms used in different parts of the thesis. These algorithms are Ant Colony Optimization (in Section 2.1.1), Genetic Algorithms (Section 2.1.2), and finally in Section 2.1.3 there is a description about the Coral Reef Optimization Algorithm.

The chapter also provides a detailed description, in Section 2.2, related to the different graph-based problems and domains used as experimental benchmarks in the thesis. These application domains are two CSP-based problems (the N-Queens problem, and the Resource-Constraint Project Scheduling Problem), Video Games and Educational Serious Games, which are described in Section 2.2.1, Section 2.2.2 and Section 2.2.3, respectively.

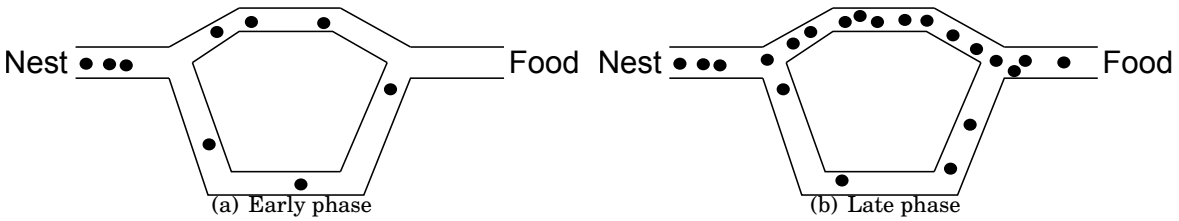
2.1 Bio-inspired algorithms

This section provides a detailed description of the different bio-inspired algorithms used in this thesis. The different algorithms used are Ant Colony Optimization algorithm, that it is based on the foraging behaviour of the ants, Genetic Algorithm, that takes inspiration in the evolutionary process of the species, and finally the Coral Reef Optimization Algorithm that it is based on the behaviour of the coral reefs.

2.1.1 Ant Colony Optimization

Ants are social insects that live in colonies. There have been many studies focus on understanding the complex behaviour that emerge from colonies. The first researcher that studied termite colonies was Eugéne Marais (1872-1936). In his book titled *The Soul of the Ant* [Marais, 1937] (published in 1937), he described his observations about the termite societies.

Following this work, Pierre-Paul Grassé [Grassé, 1959] published the mechanic of termite communication. Grassé determined a form of indirect communication called *stigmergy*. This communication was established by local interactions between individuals, and between individuals and the environment. Later, a new communication procedure called *pheromonal communication* was studied by Deneubourg in [Deneubourg et al., 1990].



(a) Early phase (b) Late phase

Figure 2.1: Representation of the double bridge experiment performed by Goss *et. al.* [Goss et al., 1989].

The ant decision making process was studied by Deneubourg *et. al.* [Deneubourg et al., 1990] and by Goss *et. al.* [Goss et al., 1989]. In the first work [Deneubourg et al., 1990], authors study the ant behaviour in a bridge with two equally long branches that separated the nest from the food. Initially, both branches were free of any pheromone, and after some time, one of the branches contained most ants. From this experiment (called *binary bridge* experiment), authors conclude that the selection of one of the branches is randomly performed when there is no pheromone in the bridge. Goss *et. al.* [Goss et al., 1989] study the ant behaviour in a bridge where the size of the branches was different. They observed that initially paths were chosen randomly with approximately the same number of ants in each branch. After some time, more ants followed the shortest branch because ants that follow the shortest branch return to the nest earlier than the other ants. This indicated that the pheromones on the shortest branch is reinforced sooner than the pheromones on the other branch. This experiment (shown in Fig. 2.1) concluded that the probability of selecting the shortest branch increases with the length ratio of the two branches.

From these studies, Marco Dorigo developed the first algorithmic model of foraging behaviour [Dorigo, 1992]. Initial studies on the foraging behaviour of ants, observed a random and chaotic activity pattern in the search of food [Dorigo and Gambardella, 1997b, Gambardella and Dorigo, 1996, Nemes and Roska, 1995]. Nevertheless, once the food source has been located, the activity of the ants becomes more organized with the ants following the same, shortest path. This self-organizing behaviour is created due to the ants that have found the food influence the rest of the ants towards the food source. This influence can be performed whereby a direct, or indirect, communication mechanism. Most ant species use an indirect communication through pheromone trails. When any ant arrives to the food source, it carries food to the nest and lays pheromones along the path. The rest of the ants will decide what path to follow based on the concentration of pheromones deposited in the path. Paths with larger pheromone concentrations have higher probabilities to be selected. If more ants follow the same path, this path will be more likely to be followed because more pheromones will be deposited in the path.

2.1.1.1 Simple Ant Colony Optimization (SACO)

Although the first ACO algorithm was initially proposed by Marco Dorigo in 1992 [Dorigo, 1992], in this chapter the chronological order has been changed to firstly describe the Simple ACO model (SACO). Both approaches have been interchanged because SACO approach can be considered as a simplification of the initial Ant System (AS) proposed by Marco Dorigo [Dorigo, 1992] and it simplifies the description of ACO algorithms. The SACO approach was proposed by Marco Dorigo and Tomas Stützle [Dorigo and Stützle, 2001] and it is based on the double bridge experiment of Deneubourg *et. al.* [Deneubourg et al., 1990] previously described.

In SACO, ants try to find the shortest path between two nodes on a graph, *G=(V,E)*, where V is the set of nodes that composes the graph and the edges, E, represent the connections between the nodes. The length of a path followed by the ant k, denoted as L^k, represents the number of hops in the path from the source to destination node.

The edges of the graph contain the pheromones deposited by the ants. The quantity of pheromone deposited in the edge connecting node i with node j in the step t is denoted by $\tau_{ij}(t)$ and initially there is not any pheromone in the graph.

In the first step, N ants are located in the source node of the graph. Ants move through the graph building a path (their own solution). For each node, the ant decides the next node that will be visited based on a probability. At step t, given an ant (k) located in node i, the probability ($p_{ij}^k(t)$) of moving to node j is defined by Eq. 2.1.

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^\alpha(t)}{\sum_{u \in \mathcal{N}_i^k} \tau_{iu}^\alpha(t)} & \text{if } j \in \mathcal{N}_i^k \\ 0 & \text{if } j \notin \mathcal{N}_i^k \end{cases} \tag{2.1}$$

Where N_i^k represents the set of feasible nodes connected to node i, for ant k starting in node i. α is a parameter that controls the influence of pheromone concentrations and $\sum_{u \in \mathcal{N}_i^k} \tau_{iu}^\alpha(t)$ is the total pheromones deposited in all the edges connecting the node i with the rest of nodes.

When any ant has found the food (i.e. it has obtained a solution), it goes back to the nest depositing pheromones on the followed path. The quantity of deposited pheromone in this path will depend on the quality of the solution found. This means that better solutions (shorter paths) will be represented with higher pheromone values and pheromones with higher values will have more chances of being selected by other ants.

The quantity of pheromone deposited by any ant (k) is inverse to the length of the solution. As it is shown in Eq. 2.2, shorter paths will have higher quantity of pheromones in the graph.

$$\triangle \tau_{ij}^k(t) \propto \frac{1}{L^k(t)} \tag{2.2}$$

Finally, the quantity of pheromone contained in each edge is computed by adding all the quantities deposited by the whole pheromone values deposited by the ants (see Eq. 2.3).

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{N} \triangle \tau_{ij}^{k}(t) \tag{2.3}$$

In order to allow the exploration of new solutions, pheromones suffer from an evaporation process. For each iteration, pheromones evaporate according to a specific evaporation rate. This evaporation can be understood as a decreasing rate in the pheromone values and it is computed as follows: $\tau_{ij}(t) = (1-\rho)\tau_{ij}(t-1)$. Therefore, the parameter ρ, $\rho \in [0,1]$, allows to control how fast the pheromones will be evaporated.

2.1.1.2 Ant System

The Ant System (AS) [Dorigo, 1992, Abbattista et al., 1995, Bonabeau et al., 1999, Dorigo et al., 1996] was the first ACO model proposed by Dorigo. This model is lightly more complex than SACO due to a change in the transition probability (defined in Eq. 2.1). In the Ant System, the probability of moving from node i to node j is defined as follows:

$$p_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{u \in \mathcal{N}_i^k} \tau_{iu}^{\alpha}(t)\eta_{iu}^{\beta}(t)} & \text{if } j \in \mathcal{N}_i^k \\ 0 & \text{if } j \notin \mathcal{N}_i^k \end{cases} \tag{2.4}$$

Where, τ_{ij} represents the quantity of pheromone deposited in the edge connecting node i to node j, whereas η_{ij} represents the heuristic value of moving from node i to node j. α and β are two parameters to balance the **exploration** and **exploitation** of the algorithm.

In any search algorithm, the **exploration** is the ability of the algorithm to search for new solutions in the unexplored regions of the solution space, whereas the **exploitation** allows a deeper analysis of a just explored region of the solution space. Therefore, the exploration allows to find different solutions for a given problem, while exploitation allows to improve found solutions.

Regarding ACO algorithm, the exploration is determined by the heuristic of the problem and the exploitation is affected by the influence of the pheromones in the ants decision. In this way, the exploitation is the ability that makes ants to follow paths (solutions) discovered by other ants, while exploration allows ants to ignore these pheromones, i.e. to ignore the found solutions, and just explore new solutions by the use of the heuristic of the problem.

In any ACO algorithm, α measures the influence of pheromones in the decision process, while β controls the influence of the heuristic function. On the one hand, if $\alpha \gg \beta$, the ants follow the paths influenced by the pheromones, and the algorithm tends to the exploitation of found solutions. On the other hand, if $\beta \gg \alpha$ the transition probability will be affected mainly by the

heuristic function and in this case the ants will tend to **explore** new solutions. Finally, note that if $\beta = 0$, the algorithm is simply the classical SACO algorithm.

AS provides some memory to the ants that is represented by N_i^k. Using AS, N_i^k includes only the set of nodes not visited and connected to node i. Note that in SACO, N_i^k contains all nodes connected to node i (visited and not visited) so it could produce loops in the path. AS ensures that each node will be only visited once.

2.1.1.3 Ant Colony System

Finally, Ant Colony System (ACS) was developed by Gambardella and Dorigo to improve the performance of AS [Bonabeau et al., 1999, Gambardella and Dorigo, 1996]. ACS incorporates a different transition rule and a different pheromone update rule. It also includes a local pheromone update procedure and candidate lists.

The transition rule is called *pseudo-random-proportional* [Gambardella and Dorigo, 1996]. It was created to balance the exploration and the exploitation processes. Any ant, k, selects the next node j to visit from, the node i, using Eq. 2.5.

$$j = \begin{cases} \arg\max_{u \in N_i^k(t)} \left\{ \tau_{iu}(t) \eta_{iu}^{\beta}(t) \right\} & \text{if } r \leq r_0 \\ J & \text{if } r > r_0 \end{cases} \tag{2.5}$$

In Eq. 2.5, r is a random value defined as $r \sim U(0,1)$ and $r_0 \in [0,1]$ is a parameter that will be used to balance the exploration and the exploitation processes. $J \in N_i^k(t)$ is a node randomly selected from the set of valid nodes, $N_i^k(t)$, using the probability defined in Eq. 2.4.

The balance between exploration and exploitation is controlled by the value of r_0 in Eq. 2.5. If $r \leq r_0$, the algorithm will exploit found solutions, whereas if $r > r_0$, the algorithm will explore new solutions. Therefore, the smaller the value of r_0, the less number of found solutions are exploited and more exploration is performed.

Regarding to the update rule, in ACS only the globally best ant is allowed to update the pheromone values of its corresponding path. This update is performed as Eq. 2.6 shows.

$$\tau_{ij}(t+1) = (1-\rho_1)\tau_{ij}(t) + \rho_1 \triangle \tau_{ij}(t) \tag{2.6}$$

This global update encourage exploitation and it is applied after all ants have constructed their corresponding solutions.

The evaporation process is also different in ACS. As can be seen in Eq. 2.6, if ρ_1 has a small value, the pheromone concentration will be slowly evaporated, therefore the best found solution will have a small influence. On the other hand, if ρ_1 has been fixed with a high value, deposited pheromones will be evaporated very fast, so the best found solution will have a strong influence.

In addition to this global update rule, ACS incorporates a local update rule that is defined by Eq. 2.7, where ρ_2 is defined in the interval $(0,1)$ and τ_0 is a small positive constant.

$$\tau_{ij}(t) = (1-\rho_2)\tau_{ij}(t) + \rho_2\tau_0 \tag{2.7}$$

Finally in ACS, $N_i^k(t)$ contains a list with the preferred nodes to be visited. Being n_l the size of the candidate list ($n_l < |N_i^k(t)|$), the n_l closest nodes (in distance or cost) to node i are included in the candidate list and ordered by increasing distance. When any ant selects a node, it will be selected the best node in the candidate list. If the candidate list is empty, then the node is selected from the rest of nodes contained in $N_i^k(t)$. The selection of non-candidate nodes can be performed by using Eq. 2.4, or by selecting the closest node to node i.

2.1.1.4 Application of ACO algorihtms

Traditionally, ACO algorithms have been applied to the Travelling Salesman Problem (TSP) [Dorigo, 1992, Dorigo and Gambardella, 1996, Guntsch and Middendorf, 2001, Stützle, 1998]. In this kind of problems a graph is used, where the nodes/vertex represents the cities and the edges stores the distances between two cities. The goal of the problem is to find the minimal length tour, that visits each city only once.

Other classical research problems are Graph Coloring [Angeline, 1995, Costa and Hertz, 1997, Dowsland and Thompson, 2008], the Quadratic Assignment Problems (QAP) [Dorigo et al., 1996, Gambardella et al., 1999, Maniezzo and Colorni, 1999] and the N-Queens problem [Solnon, 2002, Khan et al., 2009, Gonzalez-Pardo and Camacho, 2013c].

Regarding to real-world complex problems, or industrial problems, some examples are Project Scheduling Problems (PSP) [Pimont and Solnon, 2000, Merkle et al., 2002, Xing et al., 2010, Gonzalez-Pardo and Camacho, 2014b], Car Sequencing [Gottlieb et al., 2003, Morin et al., 2009], Vehicle Routing Problems (VRP) [Bullnheimer et al., 1999, Bell and McMullen, 2004], Video Games [Emilio et al., 2010, Gonzalez-Pardo et al., 2014a] or clustering [Parpinelli et al., 2002, Fernandes et al., 2008, França et al., 2008, Hamdi et al., 2008, Suomalainen, 2012], among others.

2.1.2 Evolutionary Computation

Evolutionary Computation (EC) algorithms are a population-based algorithms that search through the space of possible solutions using the Darwin's theory evolution. The key concepts of any EC algorithm is based on the natural selection and the survival of the fittest described by C. Darwin in 1859 [Darwin, 1859].

In EC algorithms, there is a population (P) composed by a set of individuals (I). Each of them represents a possible solution to the problem, and its characteristics are encoded into different *genes* that compose the *genotype*. Therefore, the genotype describes the genetic composition of an individual and the genes represent the smallest, and inheritable, units of information. Also,

individuals contain a *phenotype*, that is the representation of the genotype in the solution space. In other words, the phenotype is the expression of the traits of an individual in an environment. If we make an analogy with real world, anybody contains a *DNA* sequence with different values for the genes that composed his/her genotype, and the representation of this genotype (the *phenotype*) may result in someone who has brown eyes and curly hair.

There are different algorithms that belongs to the EC research field. The main differences amongst them are focused on how the information contained in the genotype is represented. The four fundamental families of algorithms are listed below:

- **Genetic Algorithms** (GA) are based on a linear vector representation (usually binary or integer) in the genotype [Goldberg, 1989, Holland, 1975].
- **Genetic Programming** (GP). Here the population contains a collection of programs represented by trees [Koza, 1992, Poli et al., 2008].
- **Evolutionary Programming** (EP), where the structure that is evolved is based on a finite automata [Fogel, 1962, Fogel and Fogel, 1996].
- **Evolutionary Strategies** (ES) are focused on the numeral optimization in the space of real numbers. [Rechenberg, 1965, Beyer and Schwefel, 2002].

C. Darwin said that only the fittest creatures would survive. In EC the fittest individuals are determined by the *fitness* function ($f(I_i)$, $I_i \in P$) that provides the quality of a given individual (I_i) of the population. The fitness value of any individual will determine its reproduction probability, i.e. individuals with better fitness will have more probabilities to reproduce offspring for the next generation and thus transmit its genes. Note that the term "better fitness" will depend on the goal of the problem to be solved. If the goal of the problem is to minimize the fitness function, better individuals will be those with lower fitness values. But if the goal is to maximize the function, individuals with low fitness values will be the worst individuals than those with higher values.

The general behaviour of any EC algorithm has been shown in Alg. 1. The algorithm is composed by a population (P_t) that contains n randomly initialized individuals. Once the population has been initialized, the evolution process starts.

Initially, every individual in the population is evaluated (line 4). This evaluation is performed by the fitness function. Then, one, or more, parents will be selected (line 7) from the population using the fitness values of each individual. As it was previously said, individuals with "better" fitness will have more probabilities for being selected. Once the parents have been selected, the new offspring are generated by the reproduction process (line 8). Finally, when the new population is completed (i.e. all the individuals for this new population have been created), the new population is evaluated and the process starts again.

Algorithm 1: Generic Evolutionary Algorithm

```
1  t ← 0
2  P_t ← n initialized individuals
3  while termination criteria is not satisfied do
4  |   EvaluateFitness(I_i) ∀I_i ∈ P_t
5  |   P* ← new empty population
6  |   while P* is not full do
7  |   |   parents ← selectParent(P_t)
8  |   |   offspring ← reproductionProcess(parents)
9  |   |   includeOffspring(P*,offspring)
10 |   end
11 |   t ← t + 1
12 |   P_t ← P*
13 end
```

This process is repeated until the termination criteria is satisfied. This condition is applied when:

1. The number of generations (t) has reached the maximum number of generations allowed (t_{max}). This maximum number is a parameter of the algorithm.
2. The fitness value of the population does not differ, significantly, from the fitness of the previous population. More formally: $f(P_{t+1}) - f(P_t) \leq \gamma$. Where the fitness of a population ($f(P)$) is computed as the mean fitness value for all the individuals in the population and γ is the threshold (a parameter of the algorithm).

When an EC algorithm is applied to any problem there are three problems that must be taken into account:

1. **Genotype codification**. The codification of the genotype influences the selection and reproduction processes. Also, the codification influence influence in the performance of the algorithm. Due to this thesis is not focused on EC, to carry out our different comparisons, the standard codifications (taken from literature) will be used.
2. **Selection operator**. The different operators to select the parents that will produce the new offspring are described in Section 2.1.2.1.
3. **Reproduction process**. The reproduction is composed by the *crossover* and the *mutation* operator. Both operator are described in Section 2.1.2.2.

2.1.2.1 Selection operator

The selection operator is one of the basic operators in EC algorithm and it is related to the Darwinian concept of survival of the fittest individuals. This operator is used in order to select those individuals that will produce the offspring for the next generation.

Before describing some of the selection operators, it is important to explain the concept of *selection pressure*. This concept is related to the time needed by the algorithm to produce an uniform population. The selection pressure is defined as the speed at which the best individuals will occupy the entire population by the application of this operator [Bäck, 1994, Goldberg and Deb, 1991]. If this operator has high selection pressure, the diversity of the population will decrease faster than other operators with lower selection pressure. Another problem regarding operators with high selection pressure is that this operators can converge to suboptimal (local) solution.

The most commonly used selection operators are: random, proportional, and tournament selection. The first one is the simplest selection operator where all the individuals have the same probability to be selected ($\frac{1}{n}$, where n is the population size). This kind of operator does not use any information about the fitness values of the individual, which means that the best individual has the same probability to be selected than the worst one. This operator has the lowest selection pressure from the three operators described in this section.

The second operator is the most commonly used, and it is called *Proportional Selection* [Holland, 1975]. In this kind of operator, better individuals have more probabilities to be selected. Therefore, due to the selection is directly proportional to the fitness value of the individuals, this operator has a high selection pressure. If the goal of the problem is to maximize the fitness value, the equation that defines the probability of individual i (I_i) to be selected is defined by Eq. 2.8.

$$p(I_i) = \frac{f(I_i)}{\sum_{l=1}^{n} f(I_l)} \tag{2.8}$$

Where $f(I)$ is the fitness function for individual I.

On the other hand, if the goal is to minimize the fitness value the Eq. 2.8 is not adequate because worst individuals would have more probabilities. In this case, an inverse proportional selection (defined in Eq. 2.9) is needed.

$$p(I_i) = \frac{f^*(I_i)}{\sum_{l=1}^{n} f*(I_l)} = \frac{f_{max} - f(I_i)}{\sum_{l=1}^{n} (f_{max} - f(I_l)} \tag{2.9}$$

A popular sampling method, that uses the proportional operator, is called *roulette wheel sampling*. Assuming that the fitness values are normalized and the goal is to maximize the fitness function, the roulette wheel selection is described in Alg. 2. The probability distribution

Algorithm 2: Roulette Wheel Selection

```
1 i ← 1
2 sum ← p(I_i) (using Eq. 2.8)
3 r ~ U(0,1)
4 while sum < r do
5 |   i ← i + 1
6 |   sum ← sum + p(I_i)
7 end
8 Return individual I_i as the selected individual
```

can be seen as a roulette wheel, where the size of each slice is proportional to the normalized selection probability computed by Eq. 2.8.

Finally, the last selection operator is called *tournament selection*. In this operator n_t individuals are randomly selected from the population (where $n_t \leq n$). Then the fitness values for these n_t individuals are compared and the individual with the best fitness is selected.

The selection pressure in the Tournament Selection is lower than the selection pressure in the proportional selection operator. This is produced due to the random selection of the n_t individuals. The main problems with this operator is that the parameter n_t must be fixed. If $n_t = n$ (all the population), the best individual of the population will be always selected, and the selection pressure will be increased. On the other hand, if n_t is too small, there will be more probabilities for bad individuals to be selected.

2.1.2.2 Reproduction process

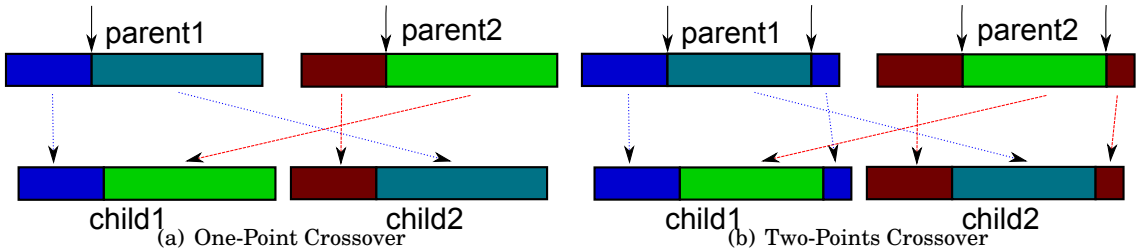
The reproduction process is the responsible of generating the new offspring once the parents have been selected. This process is composed by two operators, *crossover* and *mutation*.

The **crossover** operator generates a new child by combining the genetic material of two, or more, parents which have been selected using one of the methods described in previous section (or any other selection procedure that can be found in the literature [Fogel, 1962, Holland, 1975, Baker, 1987]). This operator is executed using a probability (p_c) that it is a parameter of the algorithm. The most commonly used crossover operator needs two parents, although there are several works that study the influence of having more than two parents [Eiben et al., 1994a].

In order to explain the different crossover techniques, lets assume that we are working with a GA algorithm (and thus the genotype is a linear vector of values) and the crossover uses only two parents.

Although, there are different crossover techniques in the literature [Srinivas and Patnaik, 1994, Schaffer and Morishima, 1987, Agrawal et al., 1994]. The simplest one is called *One-Point* crossover [Holland, 1975, Holland, 1992]. In this kind of crossover, the algorithm selects randomly one crossover point and generates the offspring interchanging the genotype blocks from the beginning of the genotype until the crossover point, and from the crossover point until the end of the genotype of both parents. This crossover technique is shown in Fig. 2.2(a). A similar (and quite popular) approach is based on two crossover points. This technique is called *Two-Points*

crossover [Bremermann et al., 1966, De Jong, 1975, Eshelman et al., 1989] and it is shown in Fig. 2.2(b).



(a) One-Point Crossover (b) Two-Points Crossover

Figure 2.2: Graphical representation of one-point, and two-points crossover for a GA algorithm.

In both techniques, the crossover points are the same for both parents. In this case, the offspring will have always the same length. There is another technique, called *Cut and slice*, where the crossover points are different in each parents. This technique ensure more diversity in the generated offspring.

Note that the examples represented in Fig. 2.2 are applied to GA algorithms. The behaviour of the crossover operator can be adapted to the particular EC algorithm considered. For example, in *Genetic Programming* (GP) [Koza, 1992, Poli et al., 2008] the genotype is represented by trees. In this case, the crossover selects randomly a node in the corresponding trees, and interchanges the selected nodes and their corresponding sub-trees (see Fig. 2.3).

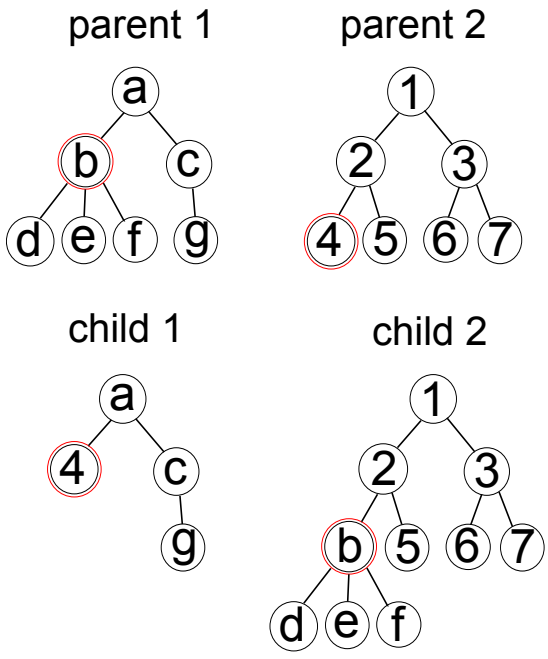


Figure 2.3: A graphical example of the crossover operator in Genetic Programming. The nodes surrounded by a red circle represent the select crossover points.

Finally, the **mutation** operator is the process of randomly changing the values of the genes in a chromosome. The main goal of this operator is to increase the genetic diversity. This operator is applied using probability (p_m) called *mutation probability*.

If the genes contains binary data, the application of this operator consists in selecting the genes randomly and the corresponding bit values are negated. In the case that the genotype contains floating-point values, the mutation operator consists in including a variation to the value of the genes [Hinterding, 1995, Michalewicz, 1996]. The general mutation operator for floating-points representations is defined in Eq. 2.10.

$$x_i'(t) = x_i(t) + \Delta x_i(t) \tag{2.10}$$

Where $x_i'(t)$ represents the mutated gene i, and $\Delta x_i(t)$ (called step size) represents the noise sampled from some probability distribution.

The mutation operator measures the exploration of the genetic algorithm and its value must be carefully fixed. High values for this probability could avoid the convergence of the algorithm, for this reason a good strategy is to fix this probability with low values to include diversity in the population and allow the algorithm to converge. For example, given a problem where the goal of the EC algorithm is to maximize the fitness function, when the mutation probability is very high the algorithm does not converge to the solution (Figure 2.4(a)). Nevertheless, assigning a lower value for this probability the behaviour of the algorithm is completely different, the algorithm converge to the optimum but it can find local solutions.

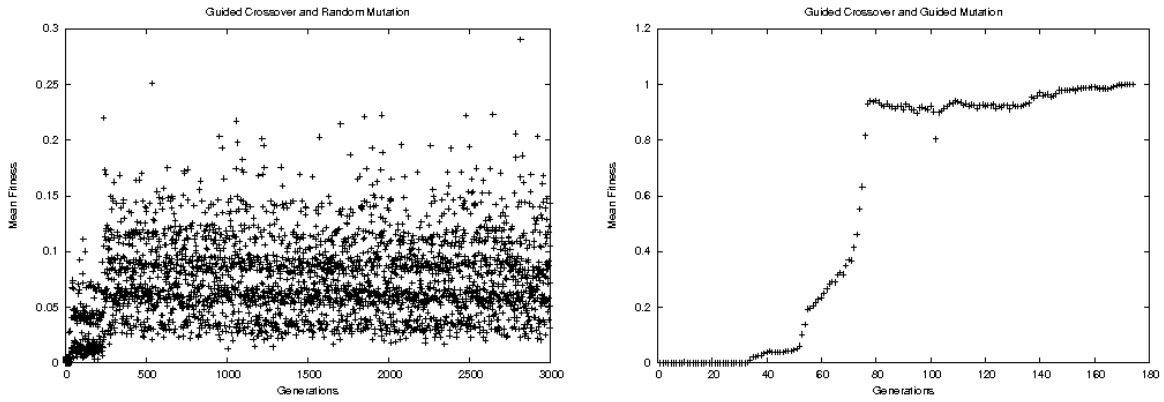


Figure 2.4: This figure shows the influence of the mutation probability in the behaviour of the EC algorithms (images extracted from [Gonzalez-Pardo et al., 2010a]) As it can be seen, when the mutation probability is high, the algorithm does not converge (Figure 2.4(a)). On the other hand, with a low mutation probability, the algorithm is able to converge to the solution of the problem (Figure 2.4(b)).

Due to all these problems, several works are focused on different parameter control techniques [Eiben et al., 1999, Barrero et al., 2010]. The parameter control consists in the self-adaptation of the algorithm, i.e. the definition of some parameter values in run-time (during the algorithm is in execution). Using parameter control, the parameters that are going to be self-adapted are encoded into the genotype of each individual, and they are affected by the reproduction procedure (mutation and crossover operators).

2.1.2.3 Genetic Algorithms

As it was previously said, in this thesis some Genetic Algorithms (GA) have been employed to compare the performance against the proposed ACO approach. GA algorithm was the first algorithm developed to simulate genetic systems. It was proposed by Fraser [Fraser, 1957b, Fraser, 1957a], and later by Bremermann [Bremermann, 1962] and Reed *et. al.* [Reed et al., 1967]. Nevertheless, GA gained popularity with the work done by Holland [Holland, 1975]. GA algorithms can be characterized by the following specifications:

- A linear vector is used in the genotype.
- The selection procedure uses a proportional selection mechanism to select the individuals.
- One-point crossover is used to produce the offspring.

2.1.2.4 Application of EC algorithms

EC algorithms can be applied to a wide range of different problems. EC can be applied from Clustering problems [Freitas, 2003, Maulik and Bandyopadhyay, 2000, Gonzalez-Pardo et al., 2010c], Travelling Salesman Problem [Goldberg and Lingle, 1985, Wang, 2014], to Information Retrieval [Martın-Bautista et al., 1999, Gonzalez-Pardo et al., 2010a, Gonzalez-Pardo and Camacho, 2011].

There are several works that apply EC algorithms to real-world complex problems such as Project Scheduling Problems (PSP) [Hartmann, 1998, Ozdamar, 1999, Gonçalves et al., 2008], Vehicle Routing Problems [Prins, 2004, Rizzoli et al., 2007, Toth and Vigo, 2001], Video Games [Kendall and Spoerer, 2004, Gonzalez-Pardo and Camacho, 2013b, Merelo et al., 2013b], [Merelo et al., 2013a, Lara-Cabrera et al., 2014], or cibersecurity [Barrero et al., 2014].

The application domains, and results, given from EC community is huge. Basically, any kind of problem related to heuristic search, optimization or hybrid solving methods (among others) have been considered by EC techniques. These large number of available results makes particularly interesting EC, and specially GA, approaches as a benchmark to compare against other computational intelligence techniques like ACO algorithms.

The application of any EC algorithm to graph-based problems does not need the creation of a graph. The requirements of the problem must be taken into account when the individuals are created (avoiding the creation of invalid individuals), or within the fitness function by penalizing the incorrect individuals. However, any kind of graph-based problem can be addressed using an EC approach, and if classical or standard problems are considered, it is possible to find published results from the community.

2.1.3 Coral Reef Optimization

Finally, the Coral Reef Optimization (CRO) is a novel metaheuristic approach based on coral reefs' formation and reproduction [Salcedo-Sanz et al., 2013b, Salcedo-Sanz et al., 2013c]. Corals are located in the reef, but the main problem of these animals is to find a place in the reef where the coral can be settled. The space in the reef is a limited resource [Genin et al., 1994] compared against the high reproduction rate of the corals. For this reason, corals must fight to obtain a place in the reef [Chadwick, 1987, Ates, 1989]. The result of this fight is that some individuals die because they can not defend the place where they are located, or because they do not find any place in the reef where they can be settled.

CRO algorithm simulates the life of the corals in the reef during different generations limited by a maximum denoted by G_{max}. For each generation, corals reproduce and fight for the space as it is shown in Algorithm 3. The algorithm is composed by the replica control procedure that avoids the best coral to populate the whole reef, the reproduction phase where new corals (solutions) are created, the *larvae* setting where the new solutions try to settle in the coral, and finally, a predation procedure by which worse corals can die and disappear from the reef.

Algorithm 3: General Coral Reef Optimization algorithm

```
1  coralInitialization
2  g ← 0
3  while g < G_max do
4  |   Reproduction phase
5  |   Evaluation of the new corals
6  |   Larvae setting
7  |   Predation process
8  |   Replica Control Procedure
9  |   g ← g + 1
10 end
11 Return the best coral in the reef
```

The reef (Λ) is defined by a set of corals ($R = \{C_1, ..., C_n\}$), where each coral represents a possible solution (I) for the modelled problem. There is, also, an evaluation function that measures the *health*, or goodness, of each coral $f(C_i) : I \Rightarrow \mathbb{R}$. This evaluation allows the comparison among corals, used in different processes like selection or reproduction.

Corals are located in a empty $\mathcal{N}x\mathcal{M}$ square grid that represents the reef and they are randomly initialized. The number of corals created in the initialization of the coral (Line 1 is defined by the size of the reef and a parameter that determines the initial population density (ρ). Thus, the initial number of corals in the reef is computed as $\mathcal{N} * \mathcal{M} * \rho$.

In the following section, the key phases from CRO algorithm are described. These phases are: reproduction phase, *larvae* setting, and predation process.

2.1.3.1 Reproduction phase

During the execution of the algorithm, the coral reef evolves generating new corals from the ones located in the reef. There are three different reproduction procedures, which are listed bellow:

1. *Broadcast Spawning (external sexual) reproduction*. This reproduction is similar to the *crossover* operator in the *EC* algorithms. Given a reef composed by C corals, the number of corals that will be reproduced using this method is defined by: $E = \{C \times P_{ext}\}$. Where P_{ext} is a parameter that defines the percentage of the corals that will participate in this procedure. Two parents randomly selected from E generates the new coral *larva* that will try to settle in the reef in the *larvae* setting procedure. Note that corals selected using *Broadcast spawning* reproduction can not be reproduce again in the same generation.

2. *Brooding (internal sexual) reproduction* consists on the generation of new larvae by copying and mutation existing corals. The number of corals that participate in this reproduction is defined by the probability P_{int}. Note that the corals that have been reproduced by previous method in the current generation can not be selected to reproduce using this method. This reproduction procedure generates 1 coral *larva* from each coral. The new coral *larva* is a copy of his parent and it is randomly mutated.

3. *Budding (asexual) reproduction*. Finally, in the third reproduction procedure all the corals are sorted according to their *health*. From this sorted list, a fraction (defined by the asexual reproduction probability P_a) of the best corals are duplicated and try to find a place in the reef.

The asexual reproduction procedure allows better corals to have more probabilities to be duplicated. This procedure could generate that the reef would be populated mainly by the best coral in the reef, and the algorithm would fall in local optima. To avoid this situation, a replica control procedure is included in the algorithm (Line 8).

This replica control procedure selects the coral with the best health in the reef and counts the number of identical corals contained in the reef. Finally, the half of those corals are removed from the reef. This replica control procedure is represented in Alg. 4.

Algorithm 4: Replica Control Procedure in the CRO algorithm

```
1  C_b ← Best Coral in the reef (Λ)
2  N ← {C_x, f(C_x) = f(C_b) ∧ ∀C_x ∈ Λ}
3  l ← |N|
4  if l > 3 then
5  |  while |N| > ⌈l/2⌉ do
6  |  |  c ← randomCoral(N)
7  |  |  if c = C_b then
8  |  |  |  Remove c of Λ
9  |  |  end
10 |  end
11 end
```

Note that the comparison performed in line 7 is needed because coral with different characteristics would have the same health. This situation can also happen in any EC algorithm, two individuals with different genotypes having the same fitness value.

2.1.3.2 *Larvae* setting

Larvae setting is the process that allows new corals to find a place into the reef. New corals are not placed in the reef once each of them are created. When a new coral larva is created it remains in a pool where the rest of the larvae will be stored for a while. Once all the reproduction process has finished, and all the larvae have been added to this pool, the larvae try to settle in the reef.

Each new coral larvae try randomly to settle in a square of the reef. If the square is empty, the coral is settled there. But if the position is not empty, the new coral will be settled only if its health is better than the coral placed in the selected location (in this case the old coral dies and it is removed from the reef). In the case that the new coral has worst value than the old coral, the new coral starts the process again looking for a new place to be settled. This process can be repeated during N_t attempts, where N_t is a parameter defined before the execution of the algorithm. If the new coral is not able to find a location in this number of attempts, the new coral dies. This process is described in Alg. 5, where N_c is the set of new *larvae* that are going to search a position in the reef.

Algorithm 5: Larvae setting procedure

```
   Data: N_c pool containing the new larvae
 1 foreach l ∈ N_c do
 2     settled ← false
 3     t ← 0
 4     while settled = false ∧ t < N_t do
 5         pos ← selectRandomPosition
 6         C ← coralInPosition(pos)
 7         if C = NULL then
 8             SettleLarvae(l, pos)
 9             settled ← true
10         else
11             if f(l) IsBetterThan f(C) then
12                 Remove(C)
13                 SettleLarvae(l, pos)
14                 settled ← true
15             else
16                 t ← t + 1
17             end
18         end
19     end
20     if t = N_t then
21         Remove(l)
22     end
23 end
```

Note that in line 11, the comparison between the coral settled in the reef and the new coral larvae is performed in terms of *IsBetterThan*. This is because this comparison depends on the goal of the problem. The problem wants to maximize the health of the corals, the comparison would be $f(l) > f(C)$. But if the goal of the algorithm is to minimize the health function, the comparison results in $f(l) < f(C)$.

2.1.3.3 Predation process

Finally, CRO algorithms incorporate a predation process that simulates the death of the corals due to external reasons. In this process, a small number of corals in the reef can die, liberating space in the reef for next generation.

Once all the larvae have been settled in the reef, all the corals are sorted by their health value. A percentage of the worst corals, defined by the parameter P_{worst}, are selected as candidates to be removed. Then each of these coral has a low probability to be removed (determined by another parameter p).

2.2 Graph-Based Problems

This section describes the different application domains used in this thesis. It has been structured into three big groups of problems. These groups are: CPS-based problems, i.e. complex problems that can be faced like any Constraint Satisfaction Problems and thus can be represented into a graph, Video Games because almost all the games contain restrictions and many of them can be modelled as graph, and finally Educational Serious Games, where the goal is to extract the student behaviours in a 3D-environment designed for learning purposes, where the interactions and gathered information can be represented using graph.

2.2.1 CSP-based Problems

Constraint Satisfaction Problems (CSP) belongs to this kind of traditional NP-hard problems with a high impact in both, research and industrial domains. The basic definition of CSPs can be stated from a set of objects (variables) that need to be assigned with a particular value, which must satisfy a set of constraints. Therefore, any CSP problem is composed by a set of variables (X) that can take binary, integer or real values from their domain (D). The values assigned to these objects must satisfy several constraints (C) (a constraint represents some kind of restriction to those values that this variable could take). Therefore, any CSP is represented with a triple (X,D,C) where $X = \{x_1,x_2,x_3,...x_n\}$ represents the set of objects that composes the problem, $D = \{d_1,d_2,d_3,...d_n\}$ is used to describe the domains that contain the different values for the objects described in X, and C represents the set of constraints that relates the objects with their values [Tsang, 1993].

There is a wide number of complex research and industrial problems that can be modelled using CSP, the main techniques, algorithms and methods obtained from this area has been applied in the last decades to real domains with an increasing level of complexity (i.e. scheduling and planning problems, energy optimization, man-power scheduling, travel and car routing

optimization, etc...) [Tsang, 1993]. This subsection provides a review of two different *NP*-hard problems that can be modelled as a CSP problem and that later have been used in this thesis. These problems are N-Queens Problem and the Resource-Constraint Project Scheduling Problem (RCPSP).

2.2.1.1 The N-Queens Problem

The N-Queens is a problem whose goal is to place N different chess-queens in a NxN chessboard in such a way queens do not attack each other, and there are not more that one queen placed in each square.

This is a classical problem proposed to undergraduate students to better understand backtracking techniques, because the complexity of this problem makes impossible its resolution by using force-brute techniques.

For the classical 8-queens problem, there are 92 solutions (one of them is shown in Fig. 2.5), although removing rotated and reflected solutions, only 12 different solutions can be found.

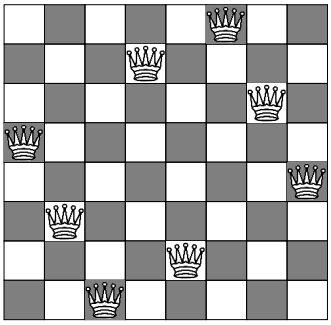


Figure 2.5: One possible solution to the classical 8-Queens Problem

Table 2.1 shows the number of different solutions related to the size of the chessboard. As it can be seen in this table, small changes in the size of the chessboard result in an explosion of the number of solutions.

2.2.1.2 Resource-Constrained Project Scheduling Problems

Resource-Constrained Project Scheduling Problem (RCPSP) is a generalization of the classical Job Shop Scheduling problem that belongs to the class of *NP*-hard optimization problems [Blazewicz et al., 1983].

In RCPSP problems there is a project composed by a set of activities $J = \{0,\ldots,n+1\}$ that needs to be processed. Note that a project, composed by n activities, has always $n+2$ activities in the set J, because activity 0 and activity $n+1$ are added to represent the initial and the final activities of the project.

Each activity can be executed in one, or more, different modes. If the activities can be executed only in one mode, the problem is called *Single-Mode* problem, but if the activities can be executed in more than one mode, the problem is called *Multi-Mode*. The modes of a given activity represent

N	Num. Solutions
1	1
2	0
3	0
4	2
5	10
6	4
7	40
8	92
9	352
10	724
11	2680
12	14200
13	73712
14	365596
15	2279184

Table 2.1: Number of different solutions for the *N-Queens* problem given the size of the chessboard.

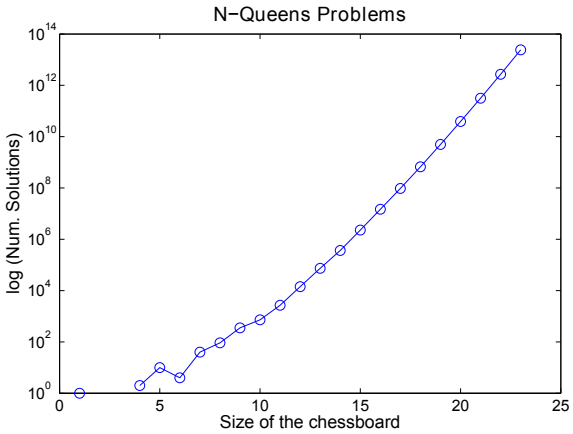


Figure 2.6: Graphical representation of the different solutions for the *N-Queens* problem based on the size of the chessboard.

different ways to execute this activity. For this reason, each mode has a duration that represents the time needed to complete the activity, and requires a set of resources.

A formal definition can be given as follows:

- The duration of the activity j executed with the mode m is denoted as d_{jm}.
- s_j denotes the time when the activity j started the execution.
- f_j represents the time when the activity has finished. Note that $f_j = s_j + d_{jm}$, because the execution of any activity cannot be interrupted.

There are two kinds of constraints that relate the different activities that compose the project. The first one is *Precedence Constraints* that force any activity not to be started before all its predecessor activities have been finished. If a given activity j has a precedence constraint with the activity i, activity i cannot be executed until activity j has finished (i.e $s_i \geq f_j$). Using these constraints, each activity j has a list with the direct predecessors (P_j) and the direct successors (S_j). Note that activity 0 is the only start activity and it has not got any predecessors, while activity $n+1$ is the only end activity and it has no successors.

The second kind of constraints is related to the different resources needed by each activity. Any project has K resources type ($K = \{1,\ldots,K\}$). When any activity j is being processed, or executed, requires r_{jk} units of the resource type $k \in K$ in each period of its duration d_j. Also, each resource type k has a limited capacity denoted as R_k at any point in time.

Fig. 2.7 shows the *Activity-On-Node* network of a single-mode RCPSP problem composed by 6 activities. The nodes of this graph represent the different activities that compose the project, whereas the edges represent the precedence constraints that relates the execution of

the activities. In this way, if activity 1 is the predecessor of activity 3, there will be an edge from node 1 to node 3 indicating that activity 3 cannot be started until activity 1 has finished its execution. Note that the duration for activities 0 and $n+1$ (in this figure called activities 0 and 7, respectively) is 0 and they do not require any resources for their execution.

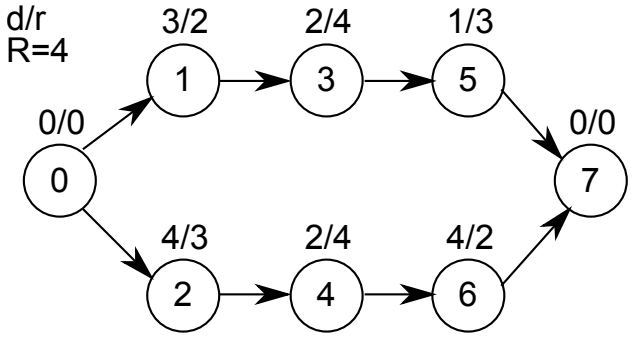


Figure 2.7: Activity-On-Node (AoN) network that describes the different activities of the project, and their precedence relations. This AoN represents a single-mode project composed by 6 activities and one resource (R) with 4 units. Each activity (i) shows its duration and the resources needed (d_i/r_i).

The goal of any RCPSP is to find a schedule for the project with the start time for all the activities, $\mathbb{S} = \{s_x \mid \forall x \in J\}$. For a given schedule, the start time is the initial time for activity 0 (s_0) and the finish time is the time for activity $n+1$ (f_{n+1}). The makespan of a schedule is defined as the difference between its finish and starting time ($f_{n+1} - s_0$). A possible schedule for the project described in Fig. 2.7, is shown in Fig. 2.8.

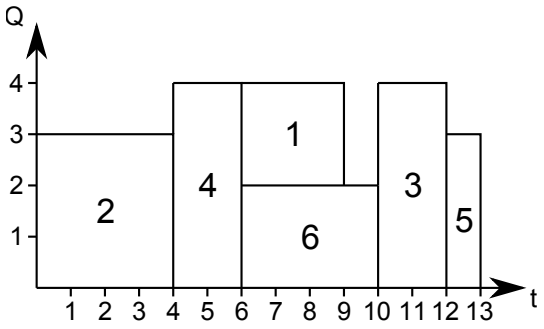


Figure 2.8: One possible schedule for the project represented in Fig. 2.7.

Any schedule is feasible, if all the precedence constraints are satisfied and for any step t, the sum of resources required for the activities in execution do not exceed the resource capacities of the project.

Let $A(t)$ be the set of activities in execution at time t, $A(t) = \{j \in J | s_j \leq t < f_j\}$. A formal model for any RCPSP problem can be defined as follows:

$$\text{Min. } f_{n+1} \tag{2.11}$$

$$f_h \leq s_j \quad j = 1,\ldots,n+1; h \in P_j \tag{2.12}$$

$$\sum_{j \in A(t)} r_{jk} \leq R_k \quad k \in K; t \geq 0 \tag{2.13}$$

$$f_j \geq 0 \quad j = 1,\ldots,n+1 \tag{2.14}$$

$$\tag{2.15}$$

The first equation (Eq. 2.11) represents the objective function that minimize the makespan of the project, i.e. to minimize the finish time for the last activity. *Precedence Constraints* are represented in Eq. 2.12 where the finish time for any activity do not have to exceed the finish time for all its predecessors. Eq. 2.13 represents the constraints regarding to the resources of the project, and the last equation (Eq. 2.14) indicates that all the activities must be scheduled.

2.2.2 Video Games

Traditionally the utilization of techniques from Artificial Intelligence (AI) area, and other sub-areas such as Computational Intelligence, has been employed in classical board games like chess, checkers, kalaha, go, othello, Tic-Tac Toe, etc... [Lucas, 2009, Reisinger et al., 2007]. However, the high impact of the video games industry has generated an increasing interest in the practical utilization of AI and CI techniques in commercial video games. Currently, there exist a wide number of international conferences that propose different competitions, or challenges, whose main goal is based on the utilization of AI and CI techniques to solve problems like: automatic solving levels, definition of autonomous Non-Player Characters (i.e. autonomous bots), automatic generation of levels, etc.

2.2.2.1 Current Video Game Platforms & Competititons

The Video Game Platorms described in this subsection are Ms. Pac-Man, Mario AI, Physical Travelling Salesman Problem (PTSP) and StarCraft. These platforms are currently used in different international challenges and competitions, so it is possible to find and compare different strategies and methods used by the research community.

Ms Pac-Man is a predator-prey arcade game, where the species, pac-man and ghost, compete, evolve and disperse simply for the purpose of seeking resources to sustain their struggle for their very existence. The game consists of a maze with paths and corridors that Pac-Man moves through collecting food pills that fill some of these paths (see Fig. 2.9). The aim of the game is to control Pac-Man in order to clear all the pills in the current maze and then advance to the next one. To do that, the methodologies based on AI/CI techniques that have been used for this game are mainly based on Genetic Programing and Coevolution [Martin et al., 2010, Liberatore et al., 2014].

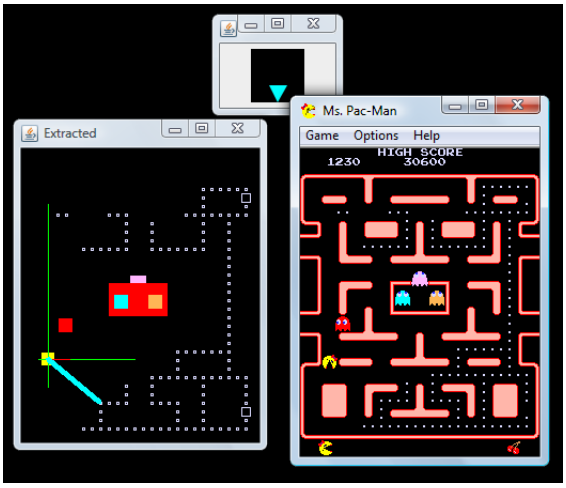


Figure 2.9: Ms Pac-Man Video Game Platform

During the game, Ms. Pac-Man is chased by four ghosts any of whom will kill Ms. Pac-Man if they are able to catch him. The ghosts behave in a non-deterministic way, which makes it hard to predict their next move although their general behaviour varies from random to very aggressive. The goal of a **Ms Pac-Man controller** is to maximise the score of the game. The different approaches performed using this game, develop controllers that takes place in the game. *Ms. Pac-Man* controllers try to maximize the average score whereas the *ghost-team controller* try to minimise the average score obtained by the Ms. Pac-Man controller that plays against it.

Another well-known Video Game used for AI/CI-based research is the Nintendo classical game called *Super Mario Bros.* [Mora et al., 2014, Togelius et al., 2013], represented by the **Mario AI** championship [Mar, 2012]. The different studies that have applied AI to this game are mainly focused on two: procedural content generation (i.e. level generations) and imitating human behaviour.

The test bed game used for the competition is a modified version of *Markus Person Infinite Mario Bros* [Moj, 2009] which is a public domain clone of Nintendo's Super Mario Bros. The character of the game (Mario) has to rescue Princess Peach that has been kidnapped by Bowser. Mario has to travel through the different two-dimensional levels avoiding obstacles and interacting with different objects to reach the final castle where the princess is located. Example of different levels are shown in Figure 2.10. Mario can move left, right, jump, and run, also depending on the objects that Mario has interacted with, he can also shoot balls of fire.

One of the main goals of the works that use this Video Game is to develop different controllers for Mario. These controllers can be based on different AI/CI techniques such as artificial evolution, evolutionary neural networks, genetic programming, fuzzy logic, temporal difference learning, human ingenuity, etc [Togelius et al., 2013].

The **Physical Travelling Salesman Problem** (PTSP) is a real-time game played on a two-dimensional map. The map has walls and obstacles, and several waypoints. The player controls a spaceship, similar to the one in the classic video game **Asteroids**. The ship can rotate using a constant angular speed, and can apply thrust in the direction that it is currently pointing. The goal is to minimize the time needed to complete the maze, the fuel consumed in the process and the received damage. The research works are focused in the development of different controllers for

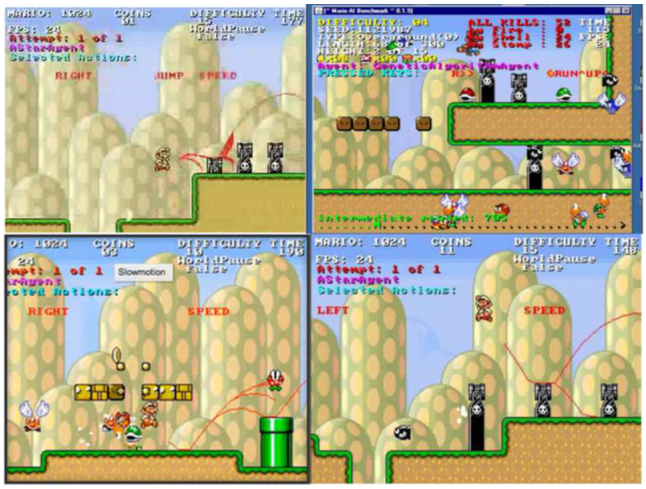


Figure 2.10: Mario AI Video Game Platform

the spaceship. The most commonly used methods are the Monte Carlo tree search, evolutionary neural networks or genetic algorithms [Coldridge and Amos, 2010].

Finally, Blizzard's **StarCraft** (Fig. 2.11) is a popular, and fun, example of real-time strategy (RTS) game. In this game, a set of races (Protoss, Zerg and Terrans) can be used to build units that have access to different technology skills, every unit works differently and requires different strategies for a player to succeed.

The enigmatic Protoss have access to powerful units, machinery and advanced technologies such as energy shields and localized warp capabilities. However, their forces have lengthy and expensive manufacturing processes, encouraging players to follow a strategy focused on the quality rather than on the quality of their units. The second raze, Zergs, possess entirely organic units and structures, which can be produced quickly at a cheap cost, but the problem is that these structures are weaker. The Terrans provide a middle ground between the other two races, providing units that are versatile and flexible. They have access to a range of more ballistic military technologies and machinery, such as tanks and nuclear weapons.

The goal of the game is to compete for resources and destroy the enemy bases. For this reason, once the game has started, the players must recollect raw material and build as quick as possible. During the game, players are constantly evolving to overcome the opponent, winning land and destroying enemies settlements.

The CIG StarCraft [Preuss et al., 2013] competitions have shown some relevant advances in the development and evolution of new StarCraft bots. Although human top Starcraft players remain unbeaten, the machines are striving to close the gap between human and artificial intelligence.

Although each race is unique in its composition, no race has an innate advantage over the other. Each specie is balanced out, so they have different strengths, powers, and abilities. But their overall strength is the same and therefore an ideal candidate to test different AI approaches like: neural networks, evolutionary algorithms, fuzzy systems, swarm intelligence and artificial immune systems [Blickle and Thiele, 1996].

Figure 2.11: Starcraft Video Game Platform

2.2.2.2 The Lemmings Video Game

The *Lemmings Game* is a popular proven NP-hard puzzle game [Cormode, 2004] that can be used as a benchmark for AI/CI algorithms. In spite of the popularity that this game obtained in the 1990s, few research has been applied to it.

Lemmings are creatures that need to be saved. In each level, Lemmings start in a specific point of the stage and must be guided to the exit point by the player (see Fig. 2.12). They live in a two-dimensional space and are affected by gravity. They start walking in a specific direction until they find an obstacle. In this case the Lemming will change the direction and walk back. In the case where the Lemming encounters a hole, it will fall down. The only two ways, considered in this thesis, by which a Lemming can die is by falling beyond a certain distance, or by falling from the bottom of the stage.

In order to make Lemmings to reach the exit point, players have a set of *skills* that must be given (not necessarily all of them) to the Lemmings. Using these skills, Lemmings can modify the environment creating tunnels, or bridges, and thus creating a new way to reach the exit.

On the one hand, there are eight different skills, with different features, that are shown in Table 2.2. Some of these skills are *Permanent* (P) which means that although the number of times that these skills can be assigned is limited, once it is assigned to the Lemming, it can use the corresponding action (i.e. Climber or Floater) several times in the same level until it reaches the exit or it dies. Other are *Temporal* (T) skills, in this case the Lemming can only use it a maximum number of times (i.e. Builder, Miner or Digger). For example, if the a Lemming has to dig in two separated locations this lemming must be assigned the Digger skill twice. Finally, there are some skills that are *Semi-Permanent* (SP) because they do not allow the Lemming to reach the exit. The semi-permanent skills are Blocker and Exploder.

In the Lemmings' world, there is a large number of materials, but all of them can be grouped in two different classes: the ones that can be modified (i.e. it can be dug) and the ones that cannot be altered. In the former type, skills like *Basher*, *Miner* and *Digger* are allowed. In the case that a Lemming is digging and finds a material that cannot be dug, the Lemming will stop digging

Skill	Description	Features
Climber	A Lemming given the climber skill can scale vertical walls	P
Floater	This skill allow the Lemming to open an umbrella if it falls beyond a high distance, avoiding its dead.	P
Exploder	The Lemming will explode after a short delay	SP
Blocker	Using this skill, a Lemming will halt and the rest of Lemming will turn around	SP
Builder	The Lemming with this skill will build a bridge of a specific length	T
Basher	To create horizontal tunnels if the environment allows it	T
Miner	This skill is similar to the previous one, but in this case the tunnel is dug in diagonal direction	T
Digger	The Lemming will dig vertically downwards until it found air or a solid material	T

Table 2.2: Lemmings skills and basic features (P: Permanent, SP: Semi-Permanent, and T: Temporal).

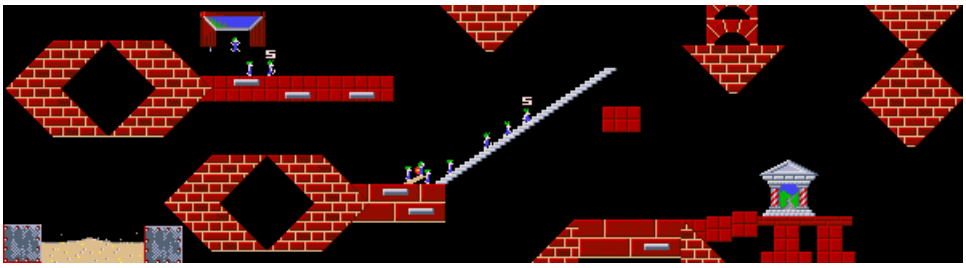


Figure 2.12: The *Lemmings* video game.

and start walking. Furthermore, each game level has its own skill configuration, where each skill can be used (i.e. assigned) a maximum number of times although it is not necessary to use all of them.

The Lemmings' game can be considered an interesting research video game problem specially for optimization algorithms. Three main objectives are necessary to optimize in this game: to save the maximum number of lemmings in each level, to minimize the use of skills needed to reach the exit of the level, and finally to find the best path that allows to save as much as possible lemmings using less skills.

The Lemmings' game have been studied in [Kendall and Spoerer, 2004]. In this work, authors applied a genetic algorithm to solve the different levels and the goal is the study of how the individuals initialization can affect to the performance of the GA.

Summarizing, the Lemmings video game provides (at least) two new interesting features. On the one hand, the video game provides different kind of terrains, that the algorithm must take into account to avoid a premature dead of the lemming, or to decide an adequate selection from the available skills. This characteristic provides an interesting "context" that should be handled by

the algorithm (for instance, by using a constraint-based modelling of the environment or a meta-heuristic to select the best skill). On the other hand, the game itself needs from the management and control of a *colony* of Lemmings. It is necessary to coordinate those lemmings to look for the best solution (which is based on a mixture of different goals).

2.2.2.3 Conclusions about the Video Games Platforms

Finally, a summary of the main features and characteristics from previous sections is presented. This summary contains the name of the video game competition, examples of frameworks, or challenges, that have been used in the corresponding competition, some of the AI and CI algorithms that have been used, and the main goals that must be reached for each competition.

- Competition: **Ms Pac-Man**
  - Platform: Software Kit [MsP, 2007], C# Kit [Flensbank and Yannakakis, 2008].
  - Algorithms: Genetic Programming [Alhejali and Lucas, 2010], Coevolution [Cardona et al., 2013].
  - Goals:
    - Ms Pac-man: to maximise the score of the game.
    - Ghost Team: to minimise the average score obtained against it by the different Ms Pac-Man controllers.
- Competition: **Platformer AI**
  - Platform: Level Generation Track [Noor Shaker and Yannakakis, 2012a], Turing Test Track [Noor Shaker and Yannakakis, 2012b].
  - Algorithms: Evolutionary neural networks [Togelius et al., 2009], Genetic Algorithm [Hou et al., 2011].
  - Goals:
    - Procedural content generation Pac-Man controllers.
    - To imitate human behaviour.
- Competition: **PTSP**
  - Platform: the PTSP Framework [Perez, 2013]
  - Algorithms: Monte Carlo Tree Search [Perez et al., 2014].
  - Goals:
    - To minimize the fuel consumed in the process.
    - To minimize the damage taken by the ship.
    - To minimize the time taken to complete the maze.

- Competition: **StarCraft**
  - Platform: AIUR for Protoss [aiu, 2011], BTHAI for Terran [bth, 2013], Skynet [bwa, 2012], BWAPI [201, 2011].
  - Algorithms: Evolutionary algorithms [Young and Hawes, 2012, Ontañón et al., 2013], Fuzzy Systems [Cadena and Garrido, 2011].
  - Goals:
    - Expansion around the map.
    - Create legions.
    - Adapt the strategy in function the enemy movements.
    - Destroy the enemy or enemies.
- Competition: **Lemmings**
  - Platform: no available platform [Gonzalez-Pardo et al., 2014a] [Gonzalez-Pardo et al., 2014b]
  - Algorithms: Ant Colony Optimization [Gonzalez-Pardo et al., 2014a] [Gonzalez-Pardo et al., 2014b, Gonzalez-Pardo and Camacho, 2013a], Genetic Algorithms [Kendall and Spoerer, 2004].
  - Goals:
    - To minimize the number of skills used to solve the level.
    - To maximize the number of lemmings that reach the exit of the levels.
    - To minimize the time needed to reach the exit.

Although a wide number of algorithms and techniques from CI (i.e. Neural Networks, Evolutionary Strategies or Fuzzy Logic) and other AI methods (from heuristic search to statistical methods) have been successfully applied, the Lemmings video game provides two new interesting features. On the one hand, the video game provides different kind of terrains, that the algorithm must take into account to avoid a premature dead of the lemmings, or to decide an adequate selection from the available skills. These characteristics provide an interesting "context" that should be handled by the corresponding algorithm (for instance, by using a constraint-based modelling of the environment, or a meta-heuristic to select the best skill). On the other hand, the game itself needs from the management and control of a *colony* of Lemmings. Therefore, it is necessary to coordinate those lemmings to look for the best solution, and the optimum solution is based on a mixture of different goals, so multi-objective algorithms could be easily applied in this domain. These new features were considered to finally decide to select this video game and not the previous ones previously described.

2.2.3 Educational Serious Games

In recent years many educational institutions have based their teaching on blended learning using gaming technologies, which combines face-to-face teaching with a high percentage of autonomous learning and online tutoring. In these years not only students, but also teachers have been facing new challenges. While students are increasingly expected to gravitate towards autonomous learning, teachers have to adopt the role of instructors and guides. This means

firstly, that teachers need to explore new ways to ensure that the expected competences are finally acquired by the students. And secondly, to make sure that learners receive the necessary guidance to succeed in their learning [Garrison and Kanuka, 2004]. One of the main trends in recent years has been the exploration and integration of Information Communication Technologies (ICTs) in autonomous learning, considered to be complementary to face-to-face learning. While face-to-face teaching refers to the general learning process which takes place in the classroom, online learning refers here to autonomous learning through Virtual Learning Environments (VLEs), that usually are supported by Learning Management Systems (LMSs). A LMS is a software application that facilitates the administration and tracking of online events and e-learning programs. It allows to manage documentation, tracking, and reporting of training programs and contents.

From the students' point of view, one interesting VLE platforms could be VWs (such as Second Life or Active Worlds), videogames and Social Network Sites (SNSs) such as Facebook or Twitter [Boyd and Ellison, 2007]. However, although SNSs are, among the students, the most popular platforms, various researchers underline that students consider SNSs a "social glue" [Madge et al., 2009] rather than a formal teaching tool [Selwyn and Grant, 2009, Lipka, 2007]. Despite the fact that the findings on the educational potentials of popular SNSs are still limited, many researchers express caution about invading a social networking space that students feel clearly is theirs in order to utilize this space for teaching purposes [Baran, 2010, Madge et al., 2009, Selwyn and Grant, 2009, Usluel and Mazman, 2009].

Although VWs have been used in different domains such as Economy [Sinrod, 2007] or E-Commerce [Talbot, 2007], the most popular are related to massively-multiplayer on-line games, such as World of Warcraft or Doom [Bellotti et al., 2008]. VWs can be used as a new powerful instrument for instruction and education, where characteristics like persistence allow continuous and growing social interactions, which can be the basis for collaborative education [Ritzema and Harris, 2008]. However, there are some characteristics that have not been fully explored in this kind of environments. Due to the immersive characteristics of VWs and the possibility to acquire automatically complex data from users (avatars) such as their current position, conversations, or operations performed in the virtual world (building objects, results of script compiling, etc...), it is possible to design real experiments where the behaviour of a set of avatars can be analyzed. Some of the previously described features can be extracted and analyzed in a real scenario [Gonzalez-Pardo et al., 2014c, Gonzalez-Pardo et al., 2010b, Berns et al., 2013].

From this thesis perspective, Virtual Worlds (VW) provide a real 3D environment that can be used as a fictional virtual world where people with different interests and skills can interact, share or cooperate in a wide range of activities [Sinrod, 2007, Talbot, 2007]. Users can interact in the VW with other users or with the elements (objects) contained in the world through avatars that represent themself (see Fig. 2.13). These interactions, and activities, can be modelled (for example using a graph-based approach) to later apply algorithms (i.e. those from Computational Intelligence previously described) to detect behavioural patterns [Gonzalez-Pardo et al., 2014c, Gonzalez-Pardo et al., 2010b, Berns et al., 2013] in avatars.

To allow the use of those different algorithms designed in this thesis, a virtual world platform, called VirtUAM (stated from Virtual world at the Universidad Autónoma de Madrid) has been used. VirtUAM is a Virtual World platform that has been designed for educational purposes. VirtUAM platform allows to automatically save the position of any avatar (tracking information), the eye-gaze of any avatar, the conversations among them (text-based information), the interactions made by them (i.e. if the avatar is flying, running or

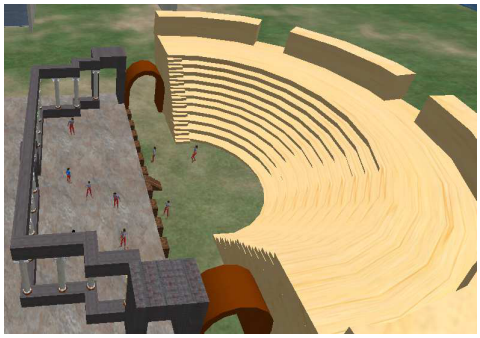


Figure 2.13: Example of a Virtual World platform.

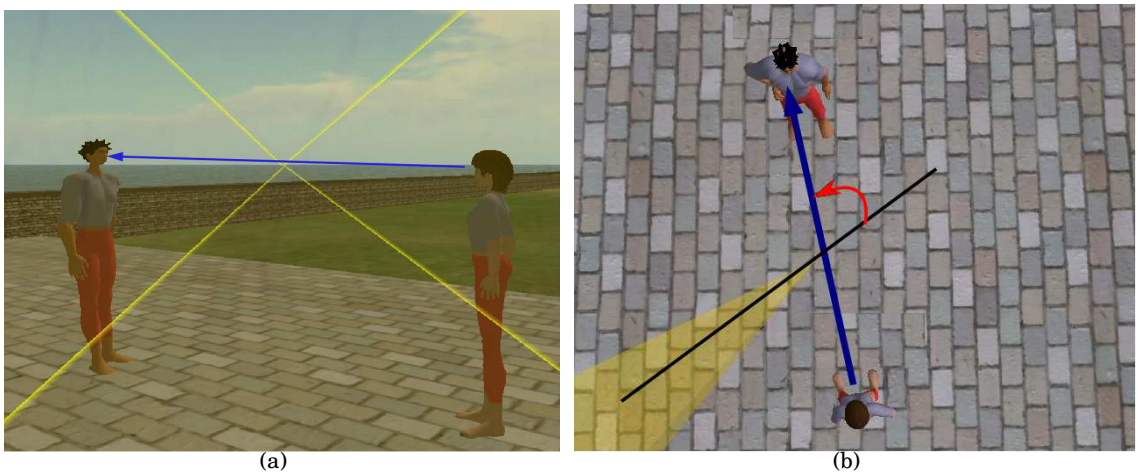
touching an object), and other actions inside the world (i.e. compiling scripts and storing the results of the compilation), etc. Once all of these information is saved, the platform allows to generate a graph-based representation. Using these data and the generated representation, different algorithms such as K-means [MacKay, 2003, Macqueen, 1967] or ACO [Abraham and Ramos, 2003, Handl and Meyer, 2007] can be used to detect behaviours in avatars.

In VirtUAM, different kinds of data are recorded from each avatar which are chat, position, gaze, interaction with the objects in the VW, and the avatar behaviour. The **chat** stores what the avatar says in the virtual world. There are two channels that any avatar could use to send messages: the public and the private channel. By using the first one any avatar near to the sender will receive the message. The meaning of ”near” will depend on the type of message sent because an avatar could *whisper* a message (in this case any avatar within a radius of 10 meters will receive the message), *say* a message within 20 meters, or *shout* a message until 100 meters. In the educational domain, the lecturer will use the public channel to give a lecture and communicate with all the students at the same time. The private channel is the other way of communication within the VW. By using this channel a bidirectional communication is created between two parts. This communication channel is a point to point communication, which means that nobody else will receive the message except the receiver.

The messages sent through both channels are relevant, because all the conversations are useful to determine if two, or more, avatars are talking about the same topic and whether this topic is related to the lecture. For that reason, all messages sent through the public or private channels are stored in order to be analyzed.

The **avatar position** is represented by the Cartesian Coordinates System, which defines the location in the VW. As a first approach, position can be used to determine what avatars are paying attention. For example, a teacher could consider that if the distance between the avatar and the teacher is greater than X meters, the student is not paying attention.

The **eye-gaze** is a calculated parameter that depends on two factors. On the one hand, it depends on the distance from the avatar to the teacher. On the other hand, it will depend on the user viewpoint, or the user camera orientation. It is important to take into account that in virtual worlds, there are two types of ”views”. The first one is the *avatar view* which is what the avatar should see. And the other view is the *user view*, or camera view, which is what the real user is seeing on the screen. This last view is the one used to compute the eye-gazing, because the goal is to determine whether the avatar is paying attention or not. With these concepts the eye-gaze is computed as follows:



(a) (b)

Figure 2.14: Representation of the eye-gazing data. Figure (a) shows how the avatar view is not the same as the user view. Figure (b) shows the same scene but from a different perspective. The angle formed between the avatar and user view, called *eye-gazing* is represented as a red arrow.

$$EyeGaze = \arccos\left(\frac{\overrightarrow{camera} * \overrightarrow{separation}}{|\overrightarrow{camera}| * |\overrightarrow{separation}|}\right) \tag{2.16}$$

Where $\overrightarrow{camera}$ is the vector defined by the user view and $\overrightarrow{separation}$ is the imaginary vector that connects the avatar with the teacher. The eye-gaze is represented in Figure 2.14. The avatar view ($\overrightarrow{separation}$) is represented by a blue arrow, and the user view ($\overrightarrow{camera}$) in Figure 2.14 is represented by a black line. The eye-gaze represents the angle formed by $\overrightarrow{camera}$ and $\overrightarrow{distance}$, and it is represented by a red arrow.

Position and gaze are recorded in three seconds intervals and this information is saved with a timestamp to facilitate its analysis. The chat is recorded when the user talks to other avatars. The timestamp for chat conversations corresponds to the moment when the conversation was initiated.

Finally, the **Avatar behaviour** is defined by those actions that avatars can perform in the VW. The set of actions taken into account are the following:

- **Move Running (MR)**. This action is registered when the avatar is running.
- **Move Flying (MF)**. This action represents the avatar while he/she is flying.
- **Move Teleporting (MP)**. Another action is teleporting. Using the teleport avatars can move immediately to any place in the VW.
- **Public Chat (XA)**. With this action any public chat conversations are represented.
- **Private Chat (XO)**. Students can chat with each other through a private channel, in this case the action is registered as XO.

- **Build Compiling (BC)**. While students are compiling or writing some functionality on scripts, this action is registered.
- **Build Touching (BT)**. This action represents the moment when the avatar is building physical objects.

While users are working in the VW, there is a procedure that stores in the Data Base (DB) the different actions performed. The result is a string composed by a sequence of actions (*MR,MF,MP,XA,XO,BC,BT*) that represents, in a sequential order, what was doing the avatar in each moment. Figure 2.15 shows both, the action string and the related actions done by the avatar. The resulting file containing these actions is called *behavioural file*.

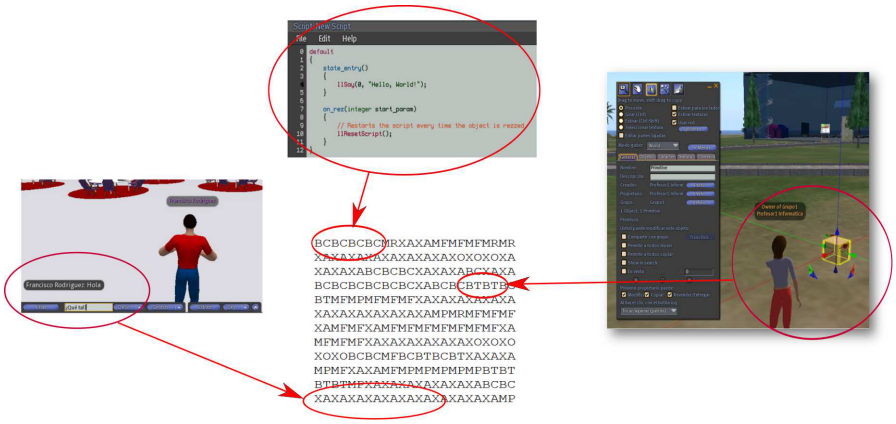


Figure 2.15: Example of different actions that avatars can perform in the VW and how this actions are represented in the behavioural file.

For this thesis, behavioural-graphs were generated by using the tracking information, eye-gaze and conversations between avatars in different experiments to analyze their behaviour [Gonzalez-Pardo and Camacho, 2014a, Gonzalez-Pardo et al., 2014c, Gonzalez-Pardo et al., 2010b, Berns et al., 2013] (see Section 5.4).

CHAPTER 3

MODELLING COMPLEX PROBLEMS BASED ON RESTRICTED GRAPHS

Any of those different problems and domains considered in this thesis can be briefly summarized as follows: any of them can be represented using some kind of restricted graph. Where the concept of "restricted" means that it is possible to connect two nodes using a restriction, or condition, that must be satisfied to solve the problem.

One straightforward example of restricted graphs is the *Constraint Satisfaction Problems*, however for this thesis work the restriction is not mandatory, so two nodes could be connected without the existence of a constraint (i.e. transition states in video games modelling). Other domains that could be considered as restricted-based graphs are video games or educational serious games, among others.

3.1 Introduction to Constraint Satisfaction Problems

Constraint Satisfaction Problems (CSP) belongs to this kind of traditional NP-hard problems with a high impact in both, research and industrial domains. The basic definition of CSPs can be stated from a set of objects (variables) that need to be assigned with a particular value, which must satisfy a set of constraints. Therefore, any CSP problem is composed by a set of variables (X) that can take binary, integer or real values from their domain (D). The values assigned to these objects must satisfy several constraints (C) (a constraint represents some kind of restriction to those values that this variable could take). Therefore, any CSP is represented with a triple (X,D,C) where $X = \{x_1,x_2,x_3,...x_n\}$ is the set of objects that composes the problem, $D = \{d_1,d_2,d_3,...d_n\}$ is the set of domains that contains the different values for the objects described in X, and C represents the set of constraints that relates the variables with their values [Tsang, 1993].

Solving any CSP involves assigning values for a set of variables in such a way that the different constraints are satisfied. The resolution of any CSP is done by providing an *assignment*

$$A = \{\, (x_1,v_1),\ (x_2,v_2),\ldots,(x_r,v_r)\,\}$$

in such a way that any variable is not assigned two different values,i.e.

$$\forall(\,(x_i,v_i),\,(x_j,v_j)\,)\in A\times A,\; x_i=x_j\Rightarrow v_i=v_j$$

and the variable assigned to the variables belongs to its domain, i.e.

$$\forall(x_i,v_i)\in A,\; v_i\in d_i$$

We can define the number of assigned variables in a assignment A as $var(A)=\{x_i \mid (x_i,v_i)\in A\}$. Any assignment is *complete* if the assignment contains all the variables of the problem, $var(A)=X$; and the assignment is *partial* otherwise.

Also, there are *consistent* assignments if all the constraints of the problem are satisfied for a given assignment, and *inconsistent* assignments if there is at least one restriction that is not true for the values in the assignment.

Finally, the resolution of any CSP consist on providing a *complete* and *consistent* assignment for the problem, i.e. an assignment that contains a value for all the variables in the problem and all the constraints are satisfied.

A simple CSP problem is the map colouring problem. The goal of this problem is to draw a graph using a finite set of colours in such a way the neighbours regions of the map have different colours. For example, Figure 3.1(a) shows a map with the different regions of Australia, and Figure 3.1(b) shows the corresponding graph for the regions.

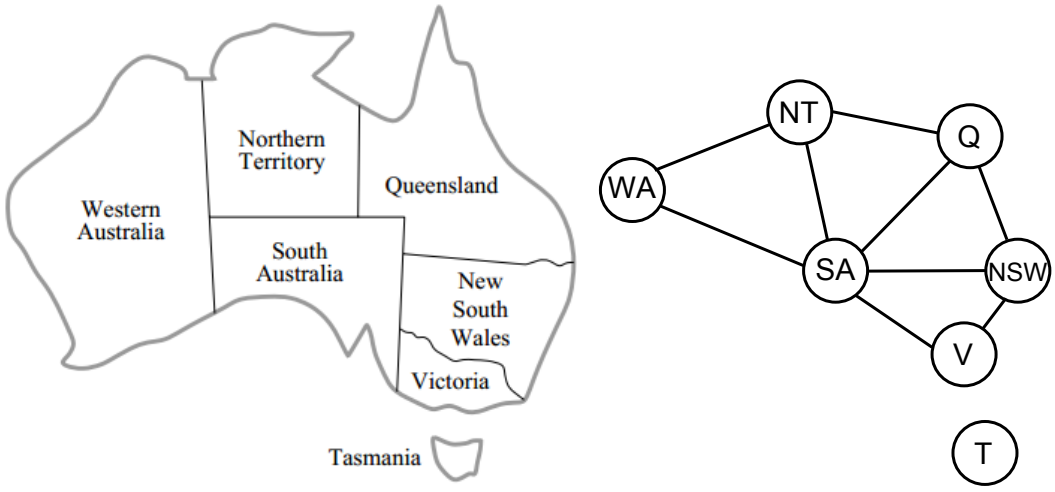


Figure 3.1: This figure shows a map with the regions of Australia (Fig 3.1(a)) and how this map is modelled into a graph (Fig 3.1(b)).

To formulate this problem as a CSP, the variables are the regions of the map ($X=\{WA,NT,SA,Q,NSW,V,T\}$), and the values for these variables are the available colours ($D=\{red,green,blue\}$). The constraint defines that two connected nodes cannot be drawn with the same colour, i.e. $D(x_i)\neq D(x_j)\;\forall x_i,x_j\in X\; such\; as\; \nexists\; E(x_i,x_j)$ where $E(x_i,x_j)$ represents an edge connecting x_i and x_j.

A possible solution to this problem could be: $\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green\}$. This solution is represented in Figure 3.2.

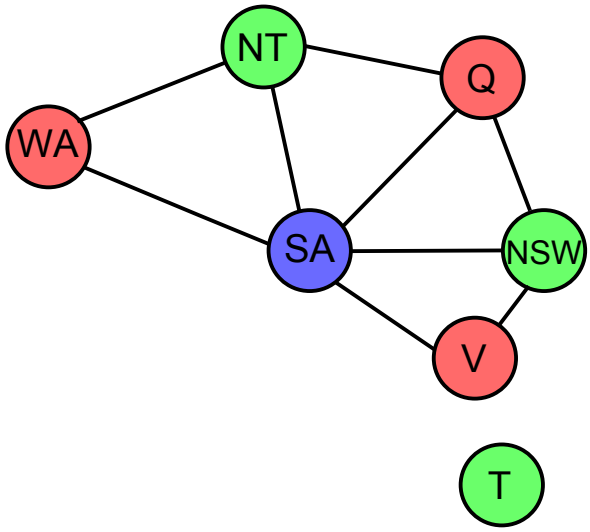


Figure 3.2: A possible solution for the graph colouring problem proposed in Figure 3.1.

There is a wide number of complex research and industrial problems that can be modelled using CSP, the main techniques, algorithms and methods obtained from this area has been applied in the last decades to real domains with an increasing level of complexity (i.e. scheduling and planning problems, energy optimization, man-power scheduling, travel and car routing optimization, etc...)[Tsang, 1993, Jabbarpour et al., 2014, Deng and Lin, 2011]. From previous domains applications, one of the critical problems to work with CSP is related to the exponential growth of the computational resources needed to solve real problems.

One way to solve any CSP problem can be done by creating a simple enumeration of all possible complete assignments and finding in the enumeration the assignment, or assignments, that are consistent. Nevertheless, this task cannot be performed in a reasonably computational time, because given a CSP composed by n variables and each variable can be given v values, there are v^k different complete assignments.

To handle and alleviate part of this combinatorial explosion in the searching process, Heuristic (informed) search has been extensively studied (i.e. BFS, A*, IDA*, Tabu Search, Hill-climbing, Simulated annealing, Local beam, Genetic Algorithms, Swarm algorithms, etc...), some of those algorithms have been specifically designed from the CSP research area. Some representative examples are: chronological backtracking (CBT), Back-jumping, conflict-directed backtracking, learning, or look-ahead techniques (i.e. Forward Checking), these algorithms are usually combined with other techniques like *constraint propagation* techniques (node consistency, arc consistency, and path consistency) to modify the constraint satisfaction problem ensuring its local consistency conditions (those related to the consistency of subsets of variables and constraints), a popular approach used to maintain the consistency in a CSP is the AC-3 algorithm. Previous searching and consistency algorithms are usually combined with a set of heuristics to improve the system performance. The searching process, these heuristics can be applied to select the next variable to be assigned (minimum width, maximum degree or maximum cardinality amongst others), or to select a value for a given variable (min-conflicts, promise, max-domain-size, weighted-max-domain-size, point-domain-size,...) [Tsang, 1993].

3.2 Heuristic Algorithms applied to Constraint Satisfaction Problems

As it has been described in previous section, the application of traditional searching algorithms to solve CSP is not an adequate approach because these problems are usually *NP-complete*. For this reason, it is really common the use of different heuristic-based searching and optimization techniques like Evolutionary Algorithms (EA) [Fogel, 1995, Eiben and Smith, 2009] and Swarm Intelligence [Engelbrecht, 2007, Blum and Merkle, 2008].

The application of GA algorithms for solving CSP have been widely used in the research community from past decades [Fonseca and Fleming, 1998, Eiben et al., 1994b]. Usually, each individual in the population represent a possible assignment that is randomly created in the initial step. The genotype for each individual has a fixed length equals to the number of variables contained in the CSP problem, and each gene is associated to each variable of the problem. Finally, the value of each gene corresponds to the value assigned to the variable encoded in the gene.

For example, given a CSP problem composed by 4 different variables (x_1,x_2,x_3,x_4) and each variable must be assigned an integer value less than 10, the genotype for the individuals in the GA will be composed by 4 genes representing each of the variables that compose the problem. Fig. 3.3 shows the genotype of an individual (I_n) and how this genotype is translated into an assignment for a CSP problem composed by 4 variables.

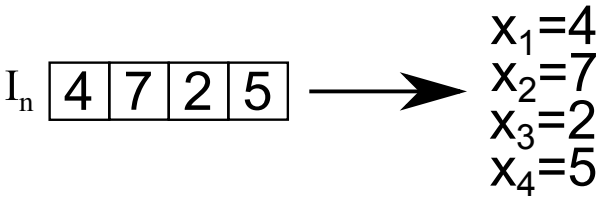


Figure 3.3: Example of the genotype representation for solving CSP.

Given a CSP composed by n variables, the individual genotype will have n genes. In this way, the gene in the position j in the genotype represents the variable x_j in the problem and the value of this gene corresponds to the value assigned to the variable x_j.

Using this approach, the crossover operator can be performed as the traditional crossover process previously described in Section 2.1.2.2. The new offspring is created by interchanging the genotypes of the selected parents (see Figure 3.4).

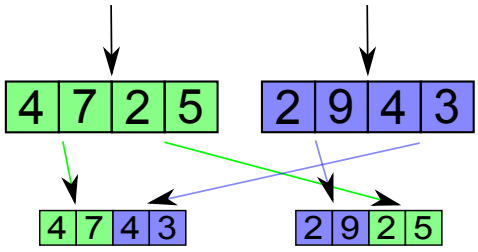


Figure 3.4: Example of the crossover operator in a standard GA for solving CSP.

The mutation can be easily implemented by changing randomly the value for the selected gene. The new value is selected from the domain values of the corresponding variable. For example, if the gene i is mutated, the value for this gene will be randomly selected from the domain values d_i, because this gene represents to the variable x_i and the possible values for this variable are defined in $d_i \in D$. Figure 3.5 shows an example of a mutation operator for a simple CSP problem, in this example the third gene is selected for mutation. This gene represents a variable whose domain values are integer numbers greater than 0 and less than 10. The new value for the gene is randomly selected from this domain value, in the example the new value is 6.

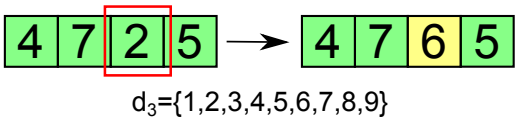


Figure 3.5: Example of the mutation operator for solving CSP.

Regarding to the crossover and mutation operation, it could be possible the generation of an invalid individual, i.e. an inconsistent assignment for the problem. In this case, the fitness function is in charge of penalizing those invalid individuals to avoid their reproduction.

Other works [Solnon, 2002, Hoseini Semnani and Zamanifar, 2012], directly related to this thesis, have applied ACO algorithms for solving CSP problems. The possible utilization of ACO algorithms to solve CSPs requires the representation of the problem as a graph where the ACO is executed (Figure 3.6).

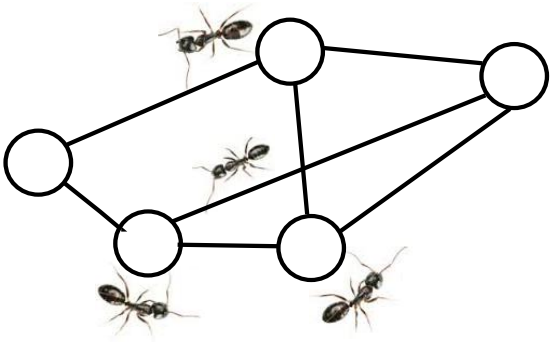


Figure 3.6: Graphical representation of a decision graph where 4 different ants are executed.

This graph, called *construction graph*, is defined as $G = (V, E)$ where V represents the nodes of the graph and E is the set of edges that connects the nodes. Roli *et. al.* proposed this construction graph for solving CSP problems [Roli et al., 2001] and they establish a full connected graph where the nodes are pairs $\langle variable, value \rangle$.

When any ant is located in a specific node, it probabilistically selects the next node from its feasible neighbourhood. Authors defined the neighbourhood as the set of $\langle variable, value \rangle$ such that the variable has not yet been assigned a value. In this way, the neighbourhood makes that any variable is only assigned once.

Although this approach has been applied in binary CSP problems, i.e. CSP problems with binary variables, several authors have used this approach in non binary CSP problems. The application of the mentioned approach in non-binary CSP, such as *N-Queens Problem*, provides some problems that have motivated this thesis.

3.3 Representation of a CSP-based problem: the *N-Queens Problem*

The *N-Queens* problem can be modelled as a CSP problem, where the variables, X, represent the set of queens, the values D are the different squares of the chessboard and the set of constraints, C, defines the different constraints of the problem:

- Two queens cannot be located in the same row.
- Two queens cannot be located in the same column.
- Two queens cannot be located in the diagonal.
- Each square of the chessboard must contain at maximum of one queen.placed in one square of the chessboard.

Each queen is defined by 2 different attributes: x and y variables, that defines the position of the queen in the chessboard, and each of them can be given N different values because for placing N queens in a board, the size of board must be $N \times N$.

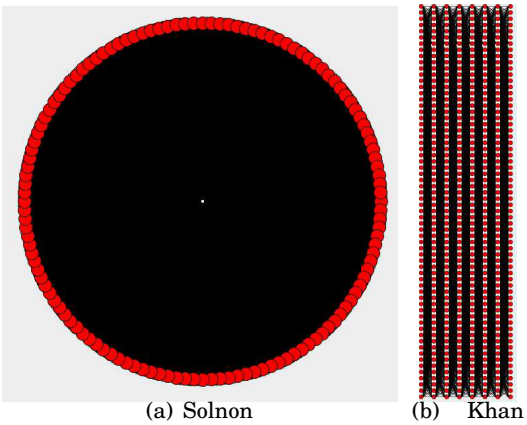
With this information, each queen is represented by $2*N$ nodes (N nodes represent the values for the variable x, and N nodes represent the values for y). As the problem is composed by N queens, the total number of nodes in the graph is $N*2*N = 2*N^2$. Using the standard approach, the resulting decision graph is full-connected. This means that each node is connected to the rest of nodes (i.e. each node is connected to $2*N^2-1$). As there are $2*N^2$ nodes in the graph, the total number of edges is $2*N^2*(2*N^2-1) = 4*N^4-2*N^2$. As it can be seen in Table 3.1, the size of the resulting graph drastically increases with the size of the problem.

One of the most popular representation, for ACO algorithms application, was defined by Solnon [Solnon et al., 2008, Solnon, 2002, Solnon, 2010] to solve the *N-Queens Problem*. In this case each queen is defined by the row and the column of the square where it is placed and the values for both variables goes from 1 to N. In this case, the graph $G = (V, E)$ is defined as follows:

$$\begin{aligned} V &= \{\langle X_i, v\rangle \mid X_i \in X \text{ and } v \in D(X_i)\} \\ E &= \left\{\left(\langle X_i, v\rangle, \langle X_j, w\rangle\right) \in V^2 \mid X_i \neq X_j\right\} \end{aligned} \tag{3.1}$$

In this case, the graph contains the same number of edges than in the standard approach ($N*(2*N) = 2*N^2$) but the number of edges in the graph is reduced. Using the approach proposed by Solnon, the resulting graph is highly connected but, it is not a full-connected graph because only those nodes representing different variables are connected. This means, two nodes $\langle X_i, v_i\rangle$ and $\langle X_j, v_j\rangle$ will be connected only if $X_i \neq X_j$. Any queen is represented by $2*N$ node, and each of these nodes will be connected to the N nodes that represent the values for the other variable for this specific queen, and the rest of nodes that represent the rest of the queens of the problems.

Therefore, each node for a given queen will be connected to $N + 2 * N * (N-1)$, where the first N refers to the nodes of the same queen that contains a different variable, and $2 * N * (N-1)$ refers to the all the nodes in the graph that represent the rest of queens. Finally, as there are $2 * N^2$ nodes, the total number of edges in the graph is $2 * N^2 * 2 * N * (N-1) = 4 * N^4 - 4 * N^3$. Table 3.1 provides the number of nodes and the number of edges for different N-Queens problems. As it can be seen in this table, Solnon's approach is the one that contains less number of edges in the graph although the resulting graph is extremely huge. Figure 3.7 shows the resulting decision graph created for the *8-Queens* problem using the Solnon's approach (Fig. 3.7(a)) and the approach proposed by Khan (Fig. 3.7(b)). In this figure, the red circles represent the nodes created by each approach, whereas the black arrows represent the different edges contained in each graph.



(a) Solnon (b) Khan et. al.

Figure 3.7: This figure provides a graphical comparison of the decision graph created for the *8-Queens* problem, using the approaches of Solnon et. al. (Fig. 4.1(a)) and Khan et. al. (Fig. 4.1(b)).

For the same problem, Khan et. al. [Khan et al., 2009] uses a different representation. In this case, the authors propose a graph composed by N layers and each layer contains N^2 nodes. Using this representation, the graph contains a layer for each queen, and each layer is composed by all the squares of the chess board. Then, each node of a layer i is connected to all the nodes of layer $i+1$ with a directional edge going from layer i to layer $i+1$, except the nodes of the last layer that are not connected to any other node. Using this approach, the resulting graph for N queens contains $N * N^2$ nodes and $(N^2 * N^2) * (N-1) \cong N^5$ edges. As Table 3.1 shows, this approach is the one that generates the biggest graph from the three approaches described in this chapter.

Using this approach, ants start in layer 1 and travel from layer i to layer $i+1$ until they reach the layer N. The action of visiting a node in layer i by any ant is similar to place the i-th queen in the square represented by the visited node.

Although, all these three approaches have revealed promising results, the size of the resulting graph is really big. This fact can produce that big problems or really complex problems are not able to be reproduced using theses approaches due to the size of the corresponding graph.

# Queens	Classical approach		Solnon 2002		Khan et. al 2009	
	# Nodes	# Edges	# Nodes	# Edges	# Nodes	# Edges
8	128	16128	128	14336	512	28672
10	200	39600	200	36000	1000	90000
50	5000	2.5×10^7	5000	2.45×10^7	125000	3.0625×10^8
75	11250	1.27×10^8	11250	1.25×10^8	421875	2.34×10^9
100	20000	4.00×10^8	20000	3.96×10^8	10^6	9.9×10^9
200	80000	6.4×10^9	80000	6.37×10^9	8×10^6	3.18×10^{11}

Table 3.1: This table compares the standard graph representation for CSP problems with the approaches of Solnon [Solnon, 2002] and Khan [Khan et al., 2009].

3.4 Swarm-based Clustering

Clustering problems consist in grouping a set of objects in such a way that objects in the same group (called a *cluster*) are more similar to each other than to those in other groups, or clusters. The objects contained in the problem are composed by a set of characteristic, or dimensions.

The complexity of a clustering problem can be determined by the dimensionality of the problem (i.e. the number of dimensions of the objects) and the number of objects to analyse.

There are several algorithms that have been designed to apply to clustering problems, such as K-Means [Macqueen, 1967, MacKay, 2003], Support Vector Machines (SVM) [Cortes and Vapnik, 1995], Neural Networks (NN) or Spectral Clustering. Also, according the scope of this thesis, GA [Gonzalez-Pardo et al., 2010c] and ACO algorithms have been applied to this purposes.

Regarding the application of GA to clustering [Gonzalez-Pardo et al., 2010c, Hruschka et al., 2009], one of the applied approach, is similar to the one used to model the *N-Queens* problem. Each gene of the genotype corresponds to one of the objects of the problem, and the value of each gene represent the number of the cluster assigned to the corresponding object.

For example, given a problem composed by 4 objects, Fig. 3.8 shows the genotype for one individual. In this figure, objects 1 and 3 (represented by the first and third gene) belongs to cluster 1, whereas clusters 2 and 3 contains the objects number 2 and 4 respectively.

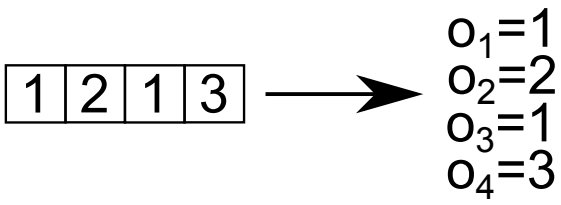


Figure 3.8: Example of the genotype representation for clustering problems.

The reproduction procedure is composed by the two operators described in Section 3.2. One-point or two-point crossover can be applied to create new offspring from two given parents (see Fig. 3.4) and the mutation operator changes randomly a gene changing the cluster where the object is classified.

Finally, when ACO algorithms are used to solve clustering problems the decision graph, where the ants are executed, must be created. This graph is highly connected and the number of nodes that will compose the graph is proportional to the number of objects in the problem and to the number of clusters where the objects must be separated.

More formally, if the problem must separate a set of objects ($O = \{o_1, o_2, \ldots, o_n\}$) in j different clusters, being $K = \{k_1, k_2, \ldots, k_j\}$ the set of labels for each cluster. The decision graph, $G = (V, e)$ is defined as follows:

$$\begin{aligned} V &= \left\{ \langle o_i, k_j \rangle \mid \forall o_i \in O \text{ and } \forall k_i \in K \right\} \\ E &= \left\{ \left(\langle o_i, k_i \rangle, \langle o_j, k_j \rangle \right) \in V^2 \mid o_i \neq o_j \right\} \end{aligned} \tag{3.2}$$

As it can be seen, this approach is very similar to the one proposed by Solnon [Solnon, 2010, Solnon, 2002] for solving CSP problems. In this approach, each object of the problem is represented by k different nodes, and each of these nodes represent the same object classified in each different cluster. Finally, two nodes are connected if they represent a different object. Figure 3.9 shows a graphical representation of the swarm-based clustering, the arrows represent the movement of any ant while it is building its own solution. In this example the first object belongs to cluster 1 whereas the second, and third, objects belong to cluster 2.

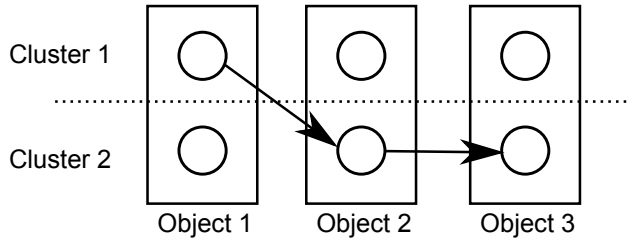


Figure 3.9: Graphical representation of a clustering problem where 3 different objects must be grouped in 2 clusters. The arrows represent the movement, and local assignation, for a given ant.

The problem showed in Figure 3.9 can be stated as a problem composed by 3 different objects ($O = \{o_1, o_2, o_3\}$) that must be grouped into 2 clusters ($K = \{k_1, k_2\}$). Nevertheless, using the classical approach, the decision graph needed to model this problem is composed by 6 nodes and 24 edges. This decision graph is shown in Figure 3.10.

Using this approach, for any problem composed by n objects that must be classified into k groups, the resulting decision graph is composed by $n * k$ nodes and $n * (n-1) * k = (n^2 * k) - (n * k)$ edges.

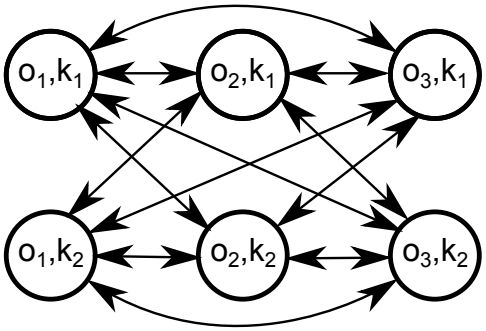


Figure 3.10: Example of the decision graph for a clustering problem composed by 3 objects (o) that must be grouped in 2 different clusters (k).

All the described approaches, can be understood as a variable assignation problem, i.e. in any CSP problem the variables must be assigned with a specific value, whereas in the clustering problem the objects must be assigned to one of the clusters available in the problem. In all the approaches described in this chapter, this assignation is encoded into each node of the decision graph. This characteristic allows the definition of a very simple behaviour of the ants, that only travel through the decision graph. Despite of this behaviour and the promising results that these approaches have revealed, the size of the resulting graph grows drastically when the size of the problem increases [Gonzalez-Pardo and Camacho, 2013c].

CHAPTER 4

A NEW ACO MODEL FOR COMPLEX GRAPH-BASED PROBLEMS

This chapter provides a detailed description of the new ACO model proposed in this thesis. This new model can be applied to any kind of problem that can be represented as a graph to execute an ACO algorithm. This chapter describes, how the problem must be modelled into a graph, what is the general behaviour of the different ants and finally, the goal of the meta-heuristic called *Oblivion Rate* is presented.

4.1 The decision graph for complex problems

This section describes how the decision graph needs to be built in order to obtain a significant reduction in the size of the resulting decision graph. The goal of this new representation is to create a smaller graphs than the ones created in the literature. As it was explained in the previous chapter, the classical decision graph created are highly connected graph composed by as many nodes as pairs $\langle variable, value \rangle$. The value for the nodes represents a specific value for a given variable (this is the case of the *N-Queens* Problem or *CSP* problems) or the label of the cluster to which the object belongs to (in the case of clustering problems).

Using the proposed model, the resulting graph is smaller because a straightforward representation of the constraint graph is used. The proposed representation [Gonzalez-Pardo and Camacho, 2013c] basically creates a node for each given variable in the problem. In this way, given a problem composed by N variables the resulting decision graph is composed by N different nodes, independently of the values that can be assigned to these variables.

The edges of the graph will connect those nodes whose values are restricted by any constraint in the problem. For example, given the nodes $\mathcal{N}_1$ (composed by variable x_1) and $\mathcal{N}_2$ (composed by x_2), it will be an edge connecting both nodes if there is, at least, one constraint involving the values of x_1 with the values of x_2. Using this approach, the number of edges is drastically reduced due to, mainly, the reduction in the number of nodes.

4.1.1 Applying the new decision graph to the *N-Queens* Problem

This subsection describes how the decision graph is created when we are dealing with the *N-Queens* problem. The goal of this problem is to place $\mathcal{N}$ queens in a NxN chessboard in such a way queens do not attack each other. In this domain, the chessboard is a two-dimensional array and each square is defined the indexes x and y that determine the position of the square into the matrix (thus x and y can vary from 0 to $\mathcal{N}-1$).

In the previous chapter, three different approaches have been described for solving the *N-Queens* problem. Given a problem composed by N queens, the standard approach [Roli et al., 2001], and the Solnon approach [Solnon, 2010], generate a decision graph composed by $2*N^2$ nodes and $4*N^4$ edges. The third approach, Khan et. al. [Khan et al., 2009], creates a decision graph that contains N^3 nodes and N^5 edges.

The size of the resulting graph using the described approaches are really big, mainly, because in all these approaches the values for the corresponding variables are encoded inside each node. For this reason, given a variable that can be assigned M different values, the variable is modelled in M different nodes. The approach that generates the biggest graph is the one proposed by Khan et. al. that creates as many chessboards as queens are contained in the problem, the goal of the ants is to allocate one queen per chessboard in such a way queens do not attack each other.

Before describing the creation of the decision graph using the proposed approach for the *N-Queens* problem, a brief formalization of the problem must be described. Given a *N-Queens* problem, there is a set of set of queens ($Q = \{q_1, q_2, \ldots, q_n\}$) that must be placed in a $N \times N$ chessboard. The position of a queen i is defined by the values of their variables x_i and y_i, that represent the row and the column of the square occupied by the queen i ($q_i = \langle x_i, y_i \rangle$). Thus, the set of queens can be defined as the pairs $\langle x, y \rangle$ that define the position of all the queens ($Q = \{\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \ldots, \langle x_n, y_n \rangle\}$).

Using the proposed approach, each variable of the problem is a *Queen*, and the task of assigning a value for each queen is traduced into assigning a value for its both coordinates, x and y. For this reason, given a problem composed by N queens, the decision graph is composed by N different nodes, that represent the N different queens that composed the problem.

As it was previously said, edges in the decision graph connect those variables related by any constraint. In the case of the *N-Queens* problem, the constraint is that the Queens do not attack each other. This constraint involve the following restrictions:

- Two queens cannot be located in the same row: $x_i \neq x_j \ \forall i \neq j \in Q$.
- Two queens cannot be located in the same column: $y_i \neq y_j \ \forall i \neq j \in Q$.
- Two queens cannot be located in the same diagonal. This restriction affects to two diagonals, the first one is defined as: $(x_i \neq x_j + k) \wedge (y_i \neq y_j + k),\ k \in [-N, N]$, whereas the second diagonal is $(x_i \neq x_j + k) \wedge (y_i \neq y_j - k),\ k \in [-N, N]$.

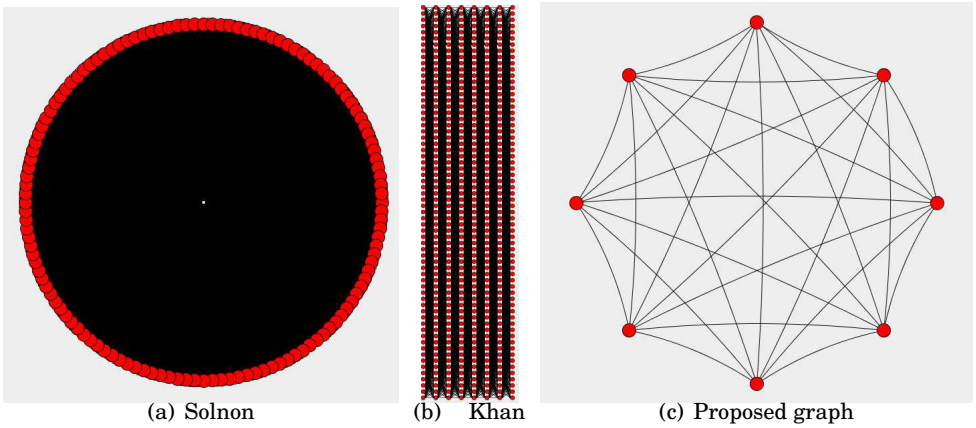
As the restrictions involve all the queens available in the problem, each node (that represent one queen) is connected to the rest of nodes available in the graph. Summarizing, for the *N-Queens* problem, the decision graph created with the proposed approach is composed by N nodes and $N*(N-1) \cong N^2 - N$.

# Queens	Solnon 2002		Khan et. al 2009		Proposed Approach	
	# Nodes	# Edges	# Nodes	# Edges	# Nodes	# Edges
8	128	15360	512	28672	8	56
10	200	38000	1000	90000	10	90
50	5000	2.478×10^7	125000	3.0625×10^8	50	2450
75	11250	1.26×10^8	421875	2.34×10^9	75	5550
100	20000	3.98×10^8	10^6	9.9×10^9	100	9900
200	80000	6.38×10^9	8×10^6	3.18×10^{11}	200	39800

Table 4.1: This table compares the model proposed in this thesis with the approaches of Solnon [Solnon, 2002] and Khan [Khan et al., 2009]. As can be seen there is an important reduction in both, the number of nodes in the resulting graph and the number of edges of the graph. Using the proposed approach applied to N-Queens problem, each node is connected to the rest of nodes (i.e. $n * (n-1)$ edges).

Table 4.1 shows a comparison among the decision graph created using the proposed approach, and the ones described in [Solnon, 2010, Khan et al., 2009]. Also Fig. 4.1 shows a graphical representation of the resulting graph for the *8-Queens* problem using the described approaches.

Using the classical approach, the resulting graph is composed by $N * x * y$ nodes and each node represents one variable (x or y) of one queen with a given value. Nevertheless, using our representation each node represents one variable for one queen and the number of nodes contained in the graph is $N * (x + y)$. The N-Queens problem is one example of those problems where some variables can be grouped and create an entity (the "Queen") that is composed by the variables x and y that defines one queen. Therefore the graph can be restructured in N nodes that represent each queen of the problem.



(a) Solnon (b) Khan et. al. (c) Proposed graph

Figure 4.1: This figure provides a graphical comparison of the decision graph created for the *8-Queens* problem, using the approaches of Solnon et. al. (Fig. 4.1(a)), Khan et. al. (Fig. 4.1(b)) and the approach proposed in this thesis (Fig. 4.1(c)).

4.1.2 Creating the decision graph for *Resource-Constraint Project Scheduling Problems (RCPSP)*

As it was previosuly described, the solution of any RCPSP consists of a schedule that describes the timesteps and ordering amongst the different activities. Moreover, if the activities have different execution modes, the solution of the problem must include the selected execution mode for each activity.

To model any RCPSP as a graph using the classical approach, it is needed the creation of two set of nodes for each activity: the first one relates the variable *startTime* and its corresponding value, while the second set is related to the execution mode for each activity. The size of the resulting graph is extremely big, due to the number of $\langle variable, value \rangle$ combinations, and the number of activities. For example, given a problem composed by X activities, each of them with M modes, the graph will have $(X * M) + 2$ nodes related to the execution modes. Note that the "+2" corresponds to the start and finish nodes of the problem, these nodes only have 1 execution mode, its duration is always 0 and they do not need any resources to be executed. Nevertheless, a trouble appears with the nodes involving the variable *startTime* (s), because this variable is defined as $s \in [0, \infty)$. It can be upper-bounded by the due date of the project, i.e. the maximum makespan allowed. Assuming that the due date of any RCPSP is d, the *startTime* for any activity is redefined as $s \in [0, d]$. For this reason, the variable *startTime* of any activity can take $d + 1$ different values, and the total number of nodes in the graph is shown in Eq. 4.1. As the graph is full-connected, each node is connected to the rest of nodes and the total number of edges in the graph is shown in Eq. 4.2.

$$|V| = (X * M) + 2 + X * (d + 1) \tag{4.1}$$

$$|E| = |V| * (|V| - 1) \tag{4.2}$$

Other option, to reduce the number of nodes tha involve the start time for each activity, consist in computing all the possible starting time for each activity taking into account the precedence constraints. The main problem with this approach is that computing all the starting time for each activity consist in searching all the possible solutions to the problem, and it is really difficult compute it, due to the complexity of RCPSP problems.

Using the proposed approach, the number of nodes that compose the graph is equal to the number of activities that compose the problem. Each node has the variable *startTime* and, if the RCPSP is a *Multi-Mode* problem, the execution mode.

In order to create the edges of the graph, the *Activity-On-Node* (AoN) network is used. The AoN is a graph where the nodes represent the activities and the edges represent the precedence relations. Figure 4.2 shows an example of the AoN of a simple RCPSP and its optimal solution is shown in Figure 4.3.

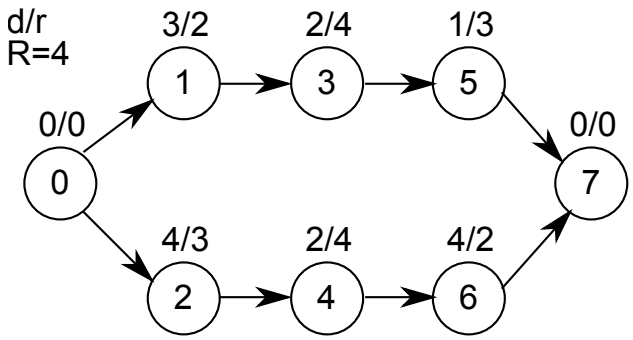


Figure 4.2: Activity-On-Node (AoN) network that describes the different activities of the project and their precedence relations. This AoN represents a single-mode project composed by 6 activities and one resource (R) with 4 units. Each activity (i) shows its duration and the resources needed (d_i/r_i).

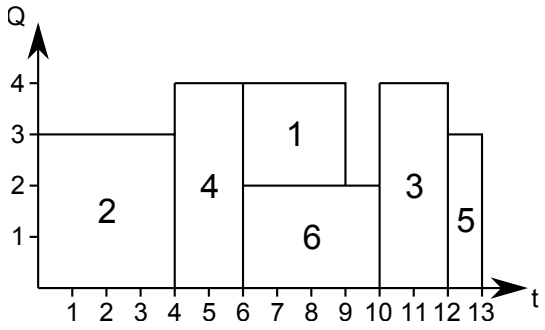


Figure 4.3: Optimal solution for the project represented in Fig. 4.2.

The application of any ACO algorithm in the AoN is not straightforward because:

1. Parallelism is not allowed. This means that the AoN showed in Fig. 4.2 only allows the execution of one of the branches of the graph either the execution of $1-3-5$ activities or the execution of the activities $2-4-6$, but ants could not execute, at the same time, activities 1 and 6 as the optimal solution shows (Fig. 4.3).

2. There are some orders that are not represented in the AoN. For example, in Fig. 4.3 once the activity 6 has finished, the activity 3 starts its execution, but these activities are not connected in the AoN because they are not predecessors.

For these reasons, the final graph is composed by the AoN and a set of new edges that solve the problems previously identified. In order to create the final graph, the list of indirect predecessors ($\mathcal{P}^*$) and indirect successors ($\mathcal{S}^*$) for each activity must be computed.

For example, in Figure 4.2 the direct successor of activity 1 is 3 ($\mathcal{S}_1 = \{3\}$). The indirect successors are composed by the direct successors of activity 3, and the direct successors of all direct successors of activity 3 (i.e. $\mathcal{S}_1^* = \{5,7\}$, 5 because it is the direct successor of activity 3 (and 3 is the direct successor of activity 1) and 7 because it is the direct successor of 5).

Once the list of indirect successors and indirect predecessors has been computed for all the activities in the project, the final graph can be created:

1. The graph contains as many nodes as activities contained in the project. Each node has two variables: the *startTime* for this activity and its *executionMode* (note that this variable is not needed if a single-mode RCPSP is modelled-9.
2. An edge from activity i to activity j is added, if both nodes are connected in the corresponding AoN. This means, if activity j is a direct successor of activity i ($j \in \mathcal{S}_i$).
3. An edge from activity i to activity j is added, if activity j is neither a direct successor nor an indirect successor of the activity i. This means, the edge from activity i to activity j will exist if $j \notin \mathcal{S}_i \wedge j \notin \mathcal{S}_i^*$.

Finally, Fig. 4.4 shows the final graph created for the RCPSP showed in Fig. 4.2.

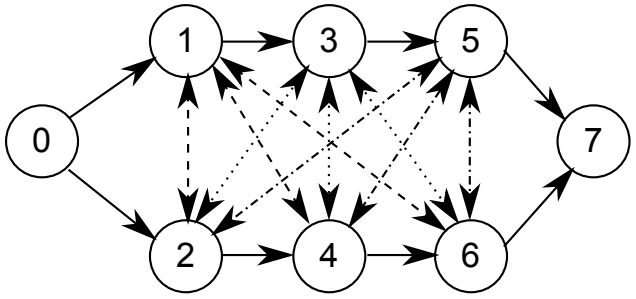


Figure 4.4: Complete graph created for the problem showed in Figure 4.2. Solid edges represent direct successors and dotted edges allow ants the execution of activities belonging to different branches of the graph.

4.1.3 The new decision graph for the *Lemmings* Video Game

The Lemmings Game can be seen as a *Constraint Satisfaction Problem* (*CSP*) where the variables, denoted as X, represents the different positions of the levels, and the possible values, D, contains the skills that lemmings can execute in each position. The set of constraints, C, is composed by the number of lemmings that must be saved, the maximum number of skills that can be applied in each level, or the different destination from a given position taking into account the applied skill (i.e. given a position the set of possible destination nodes is different whether the skill is **Builder** or **Digger**).

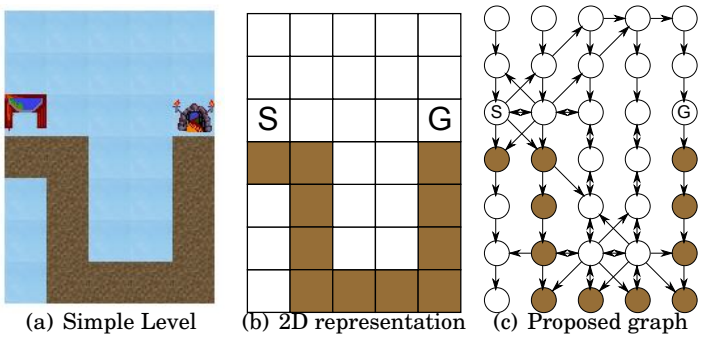
Any level of the Lemmings game is defined as a 2D grid of size $N \times M$ that represent the terrain of the level. There is also 8 different skills that can be assigned to the lemmings ($S = \{s_1, s_2, \ldots, s_8\}$) and each skill ($s_i$) can be assigned a maximum number of times t_i, ($T = \{t_1, t_2, \ldots, t_8\}$).

Using the described definition and the standard approach for modelling CSP, the problem is composed by $N * M$ variables (each position of the graph) that can be assigned using 9 different values (i.e. 8 different actions that can be applied and the standard behaviour that the lemming will execute if no skill is assigned). The decision graph contains $9 * N * M$ nodes and

$(N*M*9*(((N*M)-1)*9) \cong 81*N^2*M^2$ edges, because nodes representing the same square are not connected.

The adaptation of a Lemmings level into the simplified approach, is performed in two different phases. First, the level is represented in a two-dimensional representation that contains information about the starting point, the exit point and the terrain information of the level. In Figure 4.5 there are shown an original lemmings level (Figure 4.5(a)) and the simplification of this level into a two-dimensional representation (Figure 4.5(b)). Once the two-dimensional representation is extracted, this representation is mapped into a constraint-based graph.

The constraint-based graph contains as many nodes as squares are contained in the two dimensional representation and the edges represent the allowed movement that the ants can performed depending on the skills that can be applied. Figure 4.5(c) shows the decision graph for the lemmings level showed in Figure 4.5(a).



(a) Simple Level (b) 2D representation (c) Proposed graph

Figure 4.5: This figure shows the process of converting a level of the Lemmings game (Fig. 4.5(a)) into the proposed decision graph (Fig. 4.5(c)) using a 2D representation for the level (Fig. 4.5(b)).

4.1.4 The decision graph for clustering data from *VirtUAM*

As it was described in Section 3.4, the standard decision graph for any ACO algorithm that tries to cluster a set of x objects in k groups, or clusters, is composed by $x*k$ nodes and $(x*k)*(x-1)*k = (x^2*k^2)-(x*k^2)$ edges.

The clustering problem can be understood as a problem where x objects must be assigned a variable that can take k different values. For this reason, the resulting graph created using the proposed model is composed by x nodes. Each node contains the information of one object that needs to be separated in groups. The different nodes represent the different users that participate in the Virtual World, and the nodes contain the different information extracted from the avatars interaction (i.e. tracking position of the avatars, the messages they write in the chat, eye-gaze information, the different interactions with other avatars or users, etc.).

Although there are not any restriction amongst the objects, the resulting graph is full connected. This characteristic is needed to allow ants to visit all the nodes contained in the graph.

In the same way that in previous sections, the reduction of the graph is produced due to the reduction in the number of nodes. In the standard approach there are $n*k$ nodes whereas in the

proposed approach there are, only, n. A graphical representation of this reduction is shown in Fig. 4.6, where a clustering problem composed by 3 objects must be classified in 2 clusters. This figure shows the resulting graphs generated using the standard approach (Fig. 4.6(a)) and the proposed model (Fig. 4.6(b)).

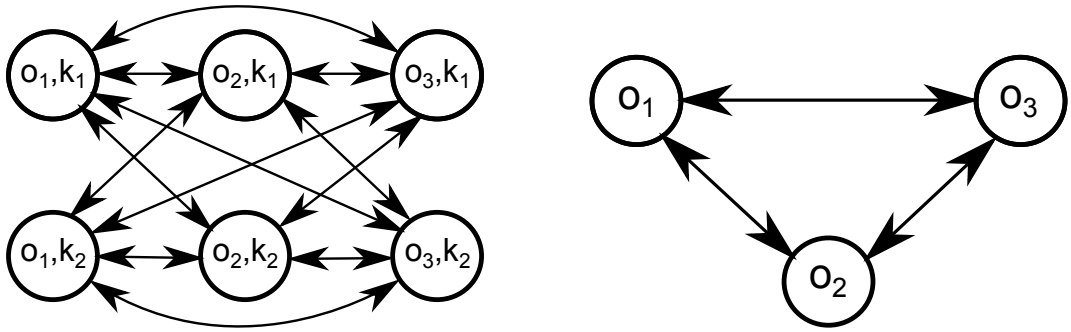


Figure 4.6: This figure shows two different representations of the resulting graph for clustering a set of 3 objects (o) in 2 different clusters (k). Fig. 4.6(a) shows the standard representation, whereas Fig. 4.6(b) shows the resulting graph using the proposed model.

4.1.5 Brief Conclusions about the Proposed Representation

The main advantage of our proposed representation is related to the complexity reduction of the graph, this allows to use ACO algorithms in more complex problems. However, the simplification of the decision graph entails two disadvantages:

1. The behaviour of the ants is more complex. In the classic approach, ants travel through the graph and the action of visiting a node represents an assignation in the problem. For example, if an ant visits the node $\langle X_i, v_i \rangle$, the ant is assigning the value v_i to the variable X_i. In the proposed model, used in this work, each time any ant visits a node, the ant must select a variable contained in the node and assign a value for this variable taking into account the different constraints. The new ants behaviour is described in Section 4.2.

2. The different combination of pairs $\langle variable, value \rangle$ instead of being represented in the number of nodes contained in the graphs, are represented in the number of pheromones that ants can deposit in the graph. Depending on the problem, the number of pheromones deposited in the graph can grow really. For this reason, a new metaheuristic for controlling the number of pheromones created in the decision graph is needed. This new metaheuristic, called *Oblivion Rate*, is addressed in Section 4.3.

4.2 Ants behaviours

The simplification of the graph entails a change in the behaviour of the ants. Using the approaches described in Section 3.2 [Khan et al., 2009, Solnon, 2002, Blazewicz et al., 1983], the ants select the next node to visit and, as the node itself contains the assignation of the variable (x_i, d_i), the ants only deposit a small quantity of pheromone in the graph and repeat the process until they finish their execution. Using the proposed representation, the behaviour of the ants is slightly more complicated because when any ant visits any node, the ant must decide what value should assign to the variable. Algorithm 6 shows this behaviour.

Algorithm 6: Ants behaviour needed in the proposed graph-representation.

```
1  EvalValList ← getEvaluatedValues(currentNode)
2  PherList ← getPheromoneInformation()
3  selectedVal ← selectValue(EvalValList,PherList)
4  updatePersonalAssignment(selectedVal)
5  D ← getPossibleNodes(currentNode)
6  if (D ≠ null) then
7  |   node ← selectNextNode(D)
8  |   currentNode ← node
9  else
10 |   resetAnt()
11 end
```

Initially, ants get the possible values for the variable stored in the current node, taking into account the heuristic function designed for the problem (Line 1). The ants, also, obtain the pheromone information of the past decision of other ants (Line 2), and select a value for the node taking into account both, the heuristic evaluation and the pheromone information (Line 3). Once the value for the variable has been selected, the ant updates its assignment (i.e. its local solution). Finally, the ant gets the possible nodes to visit, selects one of them and visits it in the next step (Line 5). Note that if there is not any possible destination, i.e. if all the connected nodes to the current node have been assigned, the ant goes back to the nest updating the pheromone information with the evaluation of the found solution (Line 10).

Note that the heuristic function, and the evaluation of the solutions represented in the pheromones, depends on the problem to be modelled. For *N-Queens* problem, the heuristic function and the pheromones determine how many queens are correctly placed if the new queen is located in the corresponding square. But for other problems, this functions are slightly more difficult.

4.2.1 Ants behaviour for the *N-Queens* problem

For the *N-Queens* problem, each node of the graph represents a queens. The task of the different ants is to determine the position of the corresponding queen in the chessboard taking into account the location of the other ants.

The position of each queen is determined by the x and y coordinates that identifies each square of the chessboard. The task of each ant is to assign a value for the corresponding variables for each node.

Algorithm 7: ACO algorithm for the *N-Queens* problem

```
   Parameter:
   Result:
1  ValList ← getAllValues(currentNode)
2  FilteredValues ← filterValues(ValList,personalAssigment)
3  if (FilteredValues ≠ null) then
4  |  PherList ← getPheromoneInformation(FilteredValues)
5  |  selectedVal ← selectValue(PherList)
6  |  updatePersonalAssignment(selectedVal)
7  |  node ← selectNextNode()
8  |  currentNode ← node
9  else
10 |  resetAnt()
11 end
```

Algorithm 7 shows the behaviour of the ants for placing the N queens. The first step is to obtain all the values and filter those invalid positions taking into account the personal solution of the ant (Line 2). In this filtering process, the different restriction of the problems are analyzed. Once the values have been filtered, if there is not any possible value the ant is reset. This situation is generated because there is not any free square due to the allocated queens in the personal solution. On the other hand, if there is, at least, one possible square where the ant can be placed, the ant gets the pheromone information (Line 4) and selects a possible value (Line 5). Finally, the ant updates its personal assignment and it visits the next unassigned queen.

4.2.2 Ants behaviour for *The Lemmings* video game

Using the described approach for solving levels for *The Lemmings* video game, the nest of the colony is located in the node that represents the starting point, and the food is located in the node that represents the exit point. From the nest, ants start building their own local solution while they travel through the graph. In order to do that, each ant executes the behaviour shown in Algorithm 8.

The first step in the algorithm corresponds to the heuristic information retrieval (line 2). In this case, ants decide whether to continue executing the current action or select a new skill. If any ant decides to apply a new skill, the ant perceive the environment (i.e. ants know the type of terrain of the surrounding nodes) and filter the available skills depending on this environment. For example, if the type of the node where the ant is placed and their surrounding are *Air*, the ant knows that the Lemming is falling and a possible skill to apply is **Floater** but not **Builder**. Once the skills have been filtered, the new skill is selected randomly taking into account the number of times that this action can be applied in the same level. This means, that given the skill **Digger** that can be used 10 times, and the skill **Miner** that can be applied 2 times, the probability of selecting **Digger** is $\frac{10}{12}$ whereas **Miner** has a probability of $\frac{2}{12}$.

Algorithm 8: ACO algorithm for the Lemmings game

Parameter: A contextual graph.
A Swarm $\mathcal{S}w$ composed by $\mathcal{L}$ Lemmings
A set of available skills $\mathcal{S}k$
Result: A path plan $\mathcal{P}$ to reach the Exit $\mathcal{G}$, from the Start $\mathcal{S}$.

```
foreach l_i ∈ L do
    HeuristicSkillList ← getSkillsUsingHeuristic()
    PheromoneValues ← getPheromoneValues()
    newAction ← selectAction(HeuristicSkillList, PheromoneValues)
    if newAction ≠ currenAction then
        if newAction canBeExecuted then
            putPheromone
            updateRemainingActions
            currentAction ← newAction
            add currentAction to P
        end
    end
    goToNextNodeAccordingTo(currentAction)
end
```

The heuristic function is in charge of evaluating the quality of any solution. This quality is composed by the number of lemmings that reach the exit (Eq. 4.3), the time needed to solve the problem (4.4) and the number of skills used (Eq. 4.5). The goal is to maximize the number of lemmings saved while the time needed to solve the problem and the number of skills used are minimized, but instead of facing this multi-objective problem, a normalized function (Eq. 4.6).

$$S(p) = TotalLemm - Blockers - ExplodedLemmings \tag{4.3}$$

$$T(p) = MaxTime - ExpendedTime \tag{4.4}$$

$$A(p) = TotalActGiv - ActionUsed(p) \tag{4.5}$$

$$Q(p) = \frac{T(p) + A(p) + S(p)}{MaxTime + TotalActGiv + TotalLemm} \tag{4.6}$$

In this case the goal is to maximize the quality of the paths (Eq. 4.6). This maximization is based on different objectives:

- To maximize the remaining time (i.e. to solve the level spending the minimum time possible).
- To maximize the number of remaining actions, this means, solving the level applying the less number of skills.
- To maximize the number of lemmings that reaches the exit. This number is computed taking into account the number of lemmings that have exploded and the lemmings that have applied the **Blocker** skill.

4.2.3 Ants behaviour for RCPSP problems

Regarding the RCPSP problems, any ant, located in activity i, can execute any activity j if the following conditions are satisfied:

1. There exists an edge from activity i to activity j.
2. All the direct and indirect predecessors of activity j have finished. I.e. $currentTime \geq f_x \ \forall x \in \mathcal{P}_j, \mathcal{P}_j^*$
3. There is enough resources to execute activity j in any of its execution modes.

The goal of this problem is to minimize the makespan of the project, i.e. to minimize the finish time of the last activity (f_{n+1}). In order to do that, the behaviour of the ants is described in Algorithm 9.

Algorithm 9: Behaviour of the ants applied to RCPSP in one step.

```
1  A ← getPossibleActivities(currentAct)
2  while A ≠ NULL do
3      act ← selectRandomActivity(A)
4      M ← getModes(act)
5      values ← evaluateModes(M)
6      PherList ← pheromoneInformation(act)
7      selectedMode ← selectMode(values,PherList)
8      updateLocalSolution(act,selectedMode)
9      currentAct ← act
10     A ← getPossibleActivities(currentAct)
11 end
```

Initially, the ants get the possible activities that can be executed (Line 1). The ants check if there is any activity with any mode that can be executed, taking into account (Line 2):

- The number of resources that are used in the current step given the local scheduled created by the ant.
- The assigned activities that have finished their execution.

If there is one or more activities that can be executed, the ants select randomly one of them. Then, the ants assign an execution mode for the selected activity (Line 7) taking into account the heuristic of the problem (Line 5) and the pheromones (Line 6). The heuristic value measures the goodness of a given mode. Given the execution mode m of an activity i, the heuristic function is defined in Eq. 4.7.

$$\tau_{im} = \gamma * d_{im} + \rho * (\sum_{q \in \mathbb{Q}} r_{iq}) \tag{4.7}$$

As can be seen in this equation, the heuristic value is composed by the time needed to finish the activity in this mode (d_{im}) and the number of resources needed in the execution ($\sum_{q \in \mathbb{Q}} r_{iq}$). The goal of the problem is to minimize the makespan of the schedule, thus modes with lower duration would be better than other modes that require more time to be completed. For this reason, γ and ρ parameters have been experimentally fixed to 0.4 and 0.6, respectively, to favour those modes with lower duration minimizing the impact of the resources.

Once the ant has selected the mode for the current activity, it updates its local solution (Line 8). This update consists of adding the current activity with the selected mode to the schedule.

Moreover, the ant needs to assign the amount of resources needed by the activity. This is a key concept in the ants behaviour, because the ant will try to execute a new activity only if it has enough resources to execute it. This check is performed in Line 10, where once the ant has updated its position in the graph, it gets the possible activities to be executed and if there is, at least, one activity, the ant repeats the process again. This fact allows ants to execute activities concurrently, i.e. two or more activities can be executed at the same time if the sum of the resources needed does not exceed the amount of unassigned resources of the project.

Finally, the pheromone values in this problem corresponds to the makespan of the found solutions. Paths represented in pheromones with lowest pheromone values, represent better solutions, because the goal of this problem is to minimize the makespan.

4.2.4 Ants behaviour for clustering problems

Regarding to clustering problems, there are two distances that define the solutions to the problem. The first distance is called *inter-cluster* distance ($\mathcal{D}_{IN}$). This distance measure the average similarity amongst the different objects that compose a cluster. The second distance measures the average similarity amongst the different clusters created in the solution. It is called *intra-cluster* distance, and it is denoted as $\mathcal{D}_{OUT}$. Fig. 4.7 shows a graphical representation of these two distances applied to a document clustering problem. This figure has been extracted from [Gonzalez-Pardo et al., 2010c].

The goal of any clustering algorithm consists in maximizing the *intra-cluster* distance whereas the *inter-cluster* distance is minimized. This is means that the clusters should be as different as possible, while the objects belonging to the same cluster must be similar.

This will be the main goal to optimize by the ants. They will build their solutions taking into account that the goal is to maximize ($\mathcal{D}_{OUT} - \mathcal{D}_{IN}$). The general behaviour of the ants is described in Alg. 10. When any ant arrives to any node, the ant gets the data stored in this node, evaluates the different clusters taking into account its own solution and the algorithm implemented (Line 2) and it gets the pheromone information (Line 3). Then the cluster, where the corresponding node is going to be classified, is selected (Line 4), the local solution of the ant is updated, and the ant selects the next node to visit.

It is important to note that the clustering algorithm used is encoded in the evaluation of the clusters. This means that the heuristic of the problem (Line 2) corresponds to the evaluation of the local solution taking into account the possible cluster and the clustering

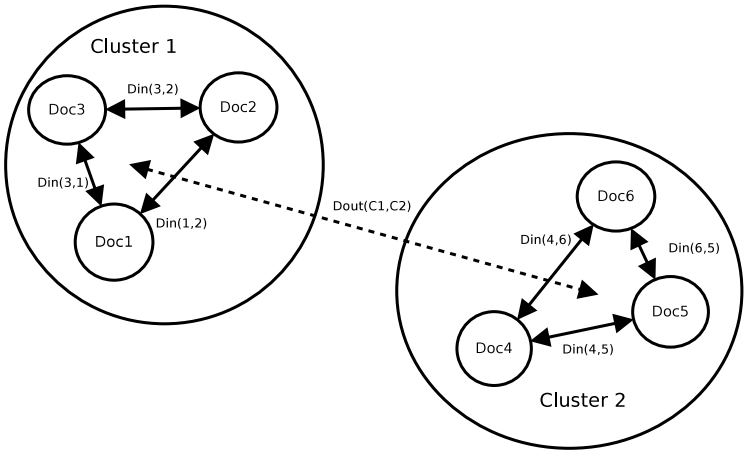


Figure 4.7: This figure shows a graphical representation about the *inter-cluster* and *intra-cluster* distances (Figure extracted from [Gonzalez-Pardo et al., 2010c]).

Algorithm 10: Behaviour of the ants applied to the clustering problem.

1 $data \leftarrow$ readNodeInfo($currentNode$)
2 $values \leftarrow$ evaluateClusters($data$)
3 $PherList \leftarrow$ pheromoneInformation($currentNode$)
4 $cluster \leftarrow$ selectCluster($values$,$PherList$)
5 updateLocalSolution($currentNode$,$cluster$)
6 $\mathcal{N} \leftarrow$ getPossibleNodes($currentNode$)
7 $currentNode \leftarrow \mathcal{N}$

algorithm (i.e. K-Means [Macqueen, 1967, MacKay, 2003] or Normalized Compression Distance [Gonzalez-Pardo et al., 2014c, Gonzalez-Pardo et al., 2010c], amongst others).

4.3 The *Oblivion Rate* metaheuristic

The reduction in the size of the decision graph affects to the behaviours of the ants, and also affects to the number of pheromones created in the graph. The main problem regarding the classical representation, described in Section 3.2, is related to the size of the resulting decision graph. In this chapter, a new model that simplifies the size of the graph has been described but the complexity derived from the number of nodes in the classical approach is converted into a fast growing in the number of pheromones created in the graph when the proposed model is used.

Pheromones, in the proposed system, contain the assignations for the different variables represented in each node. In the proposed model, the pheromones are placed in the edges of the graph, because the validity of a specific value in a node depends on the given values to the rest of nodes. For this reason, the edge connecting $Node_i$ with $Node_j$ stores all the pheromones related to the variables and values for these nodes.

Depending on the problem, it could be that the number of pheromones in the graph collapse the system. The total number of different pheromones in an edge is proportional to the size of the

domains of the variables involved in the constraints represented by the edge. That is, if $|D(var_s)|$ denotes the different values that the source variable can take, and $|D(var_d)|$ represents different values for the destination variable, the edge connecting source and destination node could store, at maximum, $|D(var_s)| * |D(var_d)|$ different pheromones.

In order to reduce the number of pheromones stored in the graph a new metaheuristic, called ***Oblivion Rate***, is included in the model with the goal of removing from the graph a subset of non-useful pheromones. The design of this metaheuristic affects directly to the system performance. Therefore, different types of ***Oblivion Rate*** functions, and their performance, were analysed in [Gonzalez-Pardo and Camacho, 2013c]. The definition and design of this heuristic is related to the problem to be solved, because it is necessary to provide an upper-bound to the maximum number of pheromones that can be created in the graph.

The ***Oblivion Rate*** metaheuristic must satisfy the following requirements:

1. The oblivion function must reduce the number of pheromones contained in the graph in order to avoid the system overload.
2. The number of pheromones must decrease gradually to allow the convergence of the system to the solution.
3. There must be always pheromones in the graph, because if in any step the system removes all the pheromones, the "*history*" of the execution is lost, and it would be similar to launch the execution from the beginning.
4. In order to converge to good solutions, the system must remember the better pheromones (those with higher pheromone values).

To look for a good Oblivion Rate function, some approaches have been analysed. The first one is a *constant function* that removes a fixed number of pheromones in each iteration. This function does not seem to be useful because it needs from the definition of a threshold. If this threshold is very high, the system would not converge because initial partial solutions would be removed and no pheromones would be deposited in the graph. On the other hand, if the threshold is too smooth, the system will not remove enough pheromones and the system would be collapsed due to the number of pheromones. Due to previous reasons, it is reasonable to use an adaptive function for determining, during the execution, the number of pheromones that the system will remember.

The Oblivion Rate is executed every step, and its behaviour is the following (Algorithm 11). The Oblivion Rate gets all the pheromones contained in the graph. In the case that the Oblivion Rate uses a negative exponential function, all the pheromones are sorted by the pheromone value. Then the Oblivion Rate computes the number of pheromones that must be forgotten (n) using the Oblivion percentage (p). Finally, the n pheromones with worst values are removed from the system. This behaviour is described in Algorithm 11.

In this thesis three different functions have been analysed. The first one, is a negative exponential function that depends on the execution step of the algorithm, the second one is a negative exponential function with a dynamic exponent. Both metaheuristics affects to the Oblivion Percentage that defines the percentage of pheromones that must be forgotten (Line 3). Finally, the last Oblivion Rate uses an adaptation of the *Coral Reef Optimization* (CRO) algorithm

Algorithm 11: Description of the Oblivion Rate metaheuristic behaviour.

```
1 P ← getAllPheromones()
2 sortPheromones(P)
3 p ← getOblivionPercentage()
4 n ← sizeOf(P)*p
5 i ← 1
6 while i ≤ n do
7 |   removeWorstPheromone(P)
8 |   i ← i + 1
9 end
```

and its behaviour is not the same as the one shown in Algorithm 11. A detail description of this metaheuristic is given in Section 4.3.3.

4.3.1 Oblivion Rate with a negative exponential function and a fixed exponent

The first Oblivion Rate function is a negative exponential function that depends on the execution step of the algorithm. Using this function when the system is finishing the execution, the majority of the pheromones would be forgotten.

The exponent of this function has been experimentally fixed. Figure 4.8 shows the function $f(step) = 1 - \frac{1}{step^x}$ with different values for the exponent x. The huge increment in the early steps when the exponent value is 1, 0.75 and 0.5 suggests that those values are not adequate because the system will converge to partial solutions. In the initial phase of the execution, the pheromone values are low (it is difficult to find the optimal solution, or solutions close to the optimal ones, in the initial steps of the algorithm) if a huge number of pheromones are removed, the system will prematurely converge to partial solutions. On the other hand, when the exponent is 0.15, or 0.25 the grow in the Oblivion Rate is much slower but in the case of 0.15, the Oblivion Rate reaches the limit of 50% pheromones removed that seems not to be enough. For previous reasons, the adaptive function for the Oblivion Rate with an exponential function has been fixed to:

$$OblivionRate(step) = 1 - \frac{1}{step^{0.25}} \tag{4.8}$$

4.3.2 Oblivion Rate with a negative exponential function and a dynamic exponent

The second function used in this thesis uses an adaptive exponent value that depends on the **saturation** of the system, taking into account the total number of possible pheromones in the system to compute the percentage of pheromones that will be removed. This metaheuristic has been applied to *N-Queens* problem and the *Resource Constraint Project Scheduling Problem*.

This Oblivion Rate applied in the step t is defined as by Eq. 4.9.

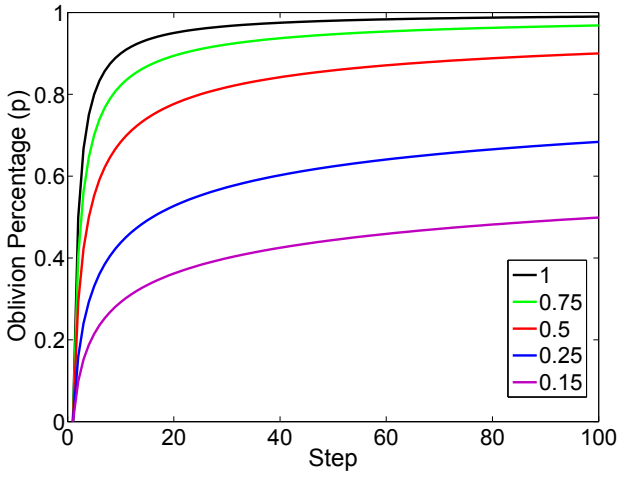


Figure 4.8: Behaviour of the Oblivion Rate when it is defined by a negative exponential function, using the number of steps and a fixed exponent in the denominator.

$$OblivionRate(t) = 1 - \frac{1}{t^{S(t)}} \tag{4.9}$$

Where $S(t)$ is the saturation of the system in the step t. This saturation takes into account the number of pheromones created in the graph ($NP(t)$) and the total number of pheromones that can be created ($MaxPher$). Eq. 4.10 defines the saturation of the system.

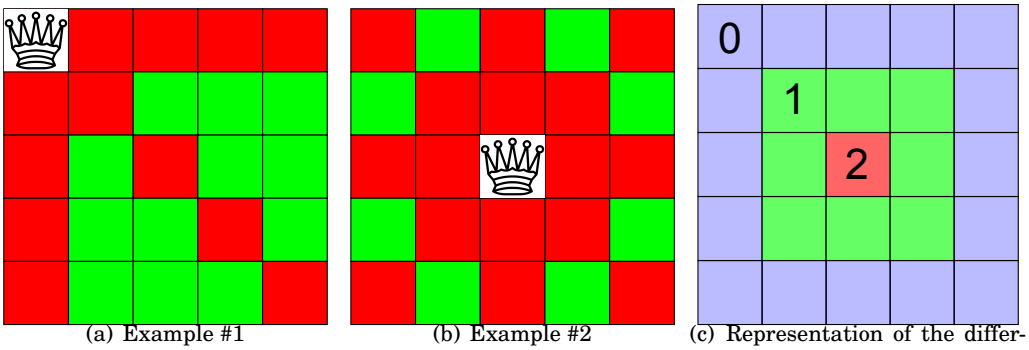
$$S(t) = \frac{NP(t)}{MaxPher} \tag{4.10}$$

In this thesis, this Oblivion Rate has been applied to the *N-Queens* problem and the *Resource-Constraint Project Scheduling Problem* (RCPSP). The following subsections describe how the maximum number of pheromones has been computed.

4.3.2.1 Oblivion Rate for the *N-Queens* problem

In this problem, the restriction determines that queens cannot attack each other. Using the proposed model, each node in the graph represents one queen and the edge connecting two nodes represent the restriction that involve both queens. This means that given two nodes, i and j, the edge connecting both nodes represents the restriction that queens i and j cannot attack each other.

The maximum number of pheromones on an edge is equal to the number of different combinations of both queens in allowed positions on the chessboard. The main problem is that the location of the first queen limits the number of possible positions where the second queen



(a) Example #1 (b) Example #2 (c) Representation of the different levels

Figure 4.9: Examples #1 and #2 represent how the number of pheromones between two queens depends on the position of the first queen. Red squares represent invalid positions for the second queen, while green squares are allowed squares. Figure 4.9(c) shows the different levels in a $5x5$ chessboard.

can be placed. This characteristic is represented in Figure 4.9(a) where there are 12 possible squares (green squares) allowed to place the other queen, whereas in Figure 4.9(b) there are only 9 available squares.

In order to compute the number of possible positions for a queen given the position of the first one (i.e. to compute the maximum number of pheromones for two queens), the concept of *level* in the chessboard have been defined. In this context, a *level* contains all the positions that if the first queen is placed in any of them, there is the same number of allowed positions for the second queen. Figure 4.9(c) shows all the levels contained in a $5x5$ chessboard. It is important to know that the number of *levels* in a chessboard of size n is $\lceil \frac{n}{2} \rceil$.

$$F(l,n) = (n-1)^2 - (n-1+2*l) \tag{4.11}$$

$$P(n) = n^2 + \sum_{i=0}^{\lceil \frac{n}{2} \rceil - 1} F(i,n) * p(i,n) \tag{4.12}$$

$$p(i,n) = \begin{cases} 1 & \text{if } n \text{ is odd and } i = \lceil \frac{n}{2} \rceil \\ \frac{2*(n-2i)+2(n-2-2i)}{n^2} & \text{otherwise} \end{cases} \tag{4.13}$$

$$MaxPher(n) = \frac{n*(n-1)}{2} * P(n) \tag{4.14}$$

The equation that defines the number of possible positions for the second queen once the first queen is placed, is shown in Eq. 4.11. In this equation, l is the level where the first queen is located, and n is the size of the chessboard. Eq. 4.12 is based on the previous equation and it defines the maximum number of combinations (pheromones) between two queens in a chessboard of size n. The first term of Eq. 4.12 (n^2) represents the possible number of destinations for the first queen. As the possible destination for the second queen depends on the location of the first one, the mathematical expectation is used to compute the total number of pheromones per level. The function $p(i,n)$ is the probability of placing a queen in the level i given a chessboard of size n. This probability is defined in Eq. 4.13. Where the fraction is used when n is even or when n is odd and $i < \lceil \frac{n}{2} \rceil$.

Eq. 4.12 returns the maximum number of pheromones between two nodes. In order to get the maximum number of pheromones in the model, Eq. 4.14 is used where $\frac{n*(n-1)}{2}$ represents the number of edges in this model applied to N-Queens problem.

Using the maximum number of pheromones (Eq. 4.14), it is possible to compute the system saturation (S) in a step (t) using the Eq. 4.10 previously described.

4.3.2.2 Oblivion Rate for *Resource-Constraint Project Scheduling Problems*

Regarding *Resource-Constraint Project Scheduling Problems* (RCPSP), the maximum number of pheromones that can be created in the model depends on the number of different jobs, j, that composed the project and also the number of modes, m, of the different jobs.

The maximum number of pheromones used in this thesis for RCPSP problems is shown in Equation 4.15. This equation represent an upper-bound of the maximum number of pheromones, because knowing exactly this number is a really difficult task.

$$MaxPher(j,m) = j * m * (j-1) * m = j * m^2 * (j-1) \tag{4.15}$$

In order to compute the maximum number of pheromones, Eq. 4.15 provides an upper bound value using the classical graph-based representation described in Section 3.2. This upper bound is computed by estimating the number of nodes, and edges, that the graph would have using the classical representation.

Using this classical representation, each job can be executed in m different ways thus the classical graph would have $j * m$ nodes. The resulting graph is highly connected and only the nodes representing the same variable are not connected. This means that each node is connected to $m * (j-1)$ nodes and the total number of edges is $j * m * (j-1) * m = j * m^2 * (j-1)$. This number of edges is our upper bound for the maximum number of pheromones in our model applied to RCPSP.

4.3.3 Oblivion Rate using an adapted CRO algorithm

Coral Reef Optimization (CRO) [Salcedo-Sanz et al., 2013a, Salcedo-Sanz et al., 2013b] [Salcedo-Sanz et al., 2013c] has been described in Section 2.1.3. The adaptation of CRO to the Oblivion Rate simplifies the behaviour of the algorithm because some of the procedures contained in CRO are not needed to control the growing of the pheromones in the model.

In this adaptation, each coral represents a pheromone created by the ants, and the health of a coral is the value of the corresponding pheromone. This thesis uses a simplified version of the CRO algorithm because the corals (or pheromones) are created by the ants and there is not any reproduction method. Therefore, there is no need to include any replica process because if any ant wants to add a coral (or pheromone) that already exists, the health of the existing coral is updated.

The graph contains a reef where the different corals (i.e. pheromones) will be stored. Initially, the coral is empty because there is not any pheromone in the graph, and no initialization procedure is needed. The ants start their execution and they place some pheromones in the reef. At this point, pheromones start with the *Larvae setting* procedure, the pheromones try to find a place in the reef in a given number of trials (determined by parameter). If the pheromone finds an empty place in the coral, the pheromone is settled. But if the position contains an old pheromone, both pheromones fights for the space. This fight consist in a comparison of the health of the corresponding pheromones. If the old pheromone has better health than the new one, the latter keeps looking for a new position. In the case where the new pheromone is better than the one found in the coral, the new pheromone is settled in the position and the old one is removed (the old coral dies). Finally, the depredation process takes place removing from the reef those corals with lower health.

In this thesis, the size of the reef is fixed using the maximum number of pheromones created in the system using the dynamic Oblivion Rate that takes into account the saturation of the system. This procedure allows us to compare the performance of both Oblivion Rates.

CHAPTER 5

EXPERIMENTAL PHASE

This chapter provides a complete description of the different experimental evaluations carried out to validate the main contributions of this thesis. Section 5.1 shows the experimental set-up and results obtained when a classical CSP problem (*N-Queens* problem) is used. The description about the different experiments that solve RCPSP problem is provided in Section 5.2. The experiments carried out to solve *The lemmings* video game are described in Section 5.3. And finally, Section 5.4 provides a description about the experimental set-up and results related to the clustering avatar behaviour problems in VirtUAM.

5.1 N-Queens Problem

This section provides the experimental set-up and the experimental results obtained for the *N-Queens* problem, previously described in Section 2.2.1.1.

This is the initial domain that was used to perform an initial study about the behaviour of the proposed model (described in Chapter 4). For this reason, these experiments have been designed to validate the proposed model and also to answer some questions related to the behaviour of the Oblivion Rate against the traditional evaporation rate, used in the literature.

As it was described in Section 4.1.1, the proposed model obtains a strong reduction in the graph complexity used to solve the *N-Queens* problem. However, this reduction affects to both, the ants behaviour and the growth of the number of pheromones created in the graph. Therefore, theses experiments will analyze the Oblivion Rate behaviour and how the application of this metaheuristic affects to the solving performance of the ACO algorithm.

5.1.1 Could the standard Evaporation Process work as the Oblivion Rate?

The main goal of the Oblivion rate is to reduce the number of pheromones contained in the graph. Although the goal of the evaporation process is to reduce the pheromone values in order to difficult worst paths to be selected, the number of pheromones in the graph does not decrease. That is, the execution of an ACO with a high evaporation rate generates pheromones with very low pheromone values, but the pheromones still remain in the system and they can finally saturate it.

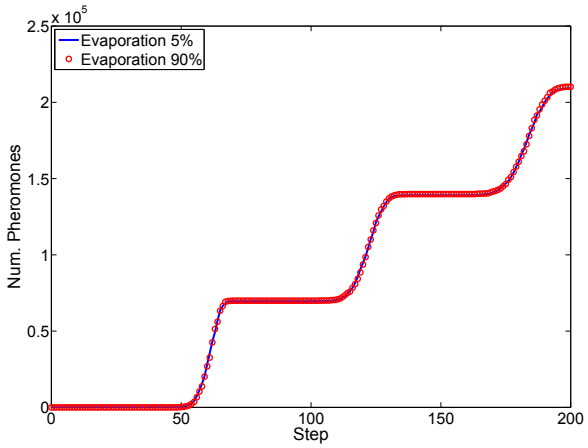


Figure 5.1: This figure shows the evolution in the number of pheromones using different evaporation rates. 100 ants try to place 25 queens in a 25×25 chessboard during 200 steps.

In order to prove this fact, Figure 5.1 shows the growth of pheromones created in the system configured with a high evaporation rate. In this experiments 100 ants try to solve the *25-Queens* problem during 200 steps. As it can be seen in this figure, the number of pheromones is not reduced. This fact emphasizes the need of using the Oblivion Rate in order to avoid the saturation of the system.

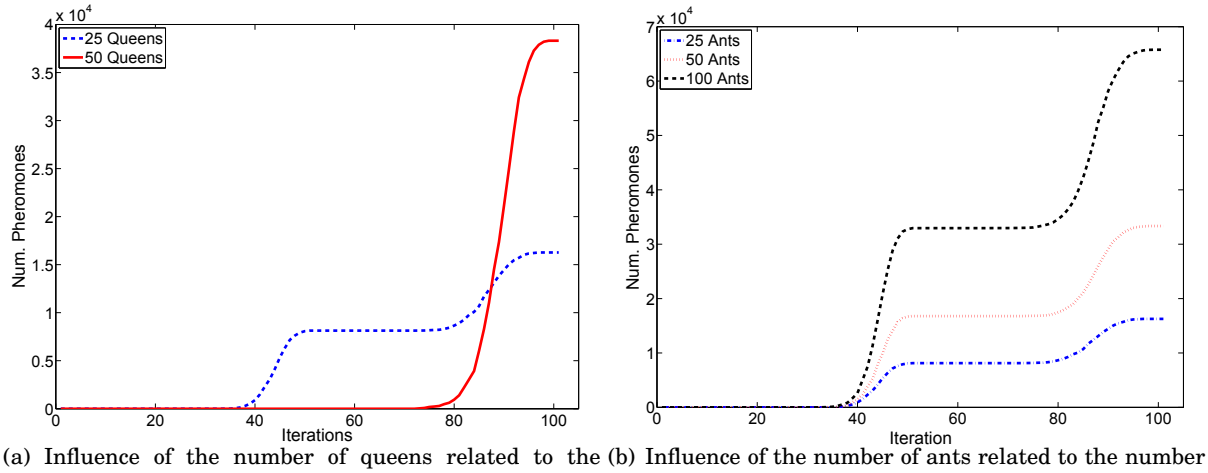
5.1.2 Is the oblivion function useful?

As it was said in Chapter 4, the reduction in the resulting graph affects to the number of pheromones that can be stored in the system. And, depending on the problem, if the number of pheromones created is very large, the system would be overloaded and the performance of the system would get worse. This is produced because the complexity reduction in the number of nodes, in our approach, produces a fast growing in the number of pheromones created in the graph.

Regarding the *N-Queens* problem, there are three factors that affects to the number of pheromones created:

1. The number of queens of the problem.
2. The number of ants in the system. The number of ants in an ACO algorithm represents the number of ”parallel” searches in the search space. If the number of ant increases, the number of searches increases, and thus, the number of pheromones deposited in the graph grows faster.
3. The number of iterations that the system is executed. When the number of iterations increases, ants expend more time searching for different solutions and then, more pheromones are deposited in the system.

Figure 5.2(a) and Figure 5.2(b) shows how the previous parameters affects to the growth of pheromones. Also can be seen that the number of pheromones is very high: 4×10^4 in the case of Figure 5.2(a) and 7×10^4 in the case of Figure 5.2(b). These pheromones congest the system and the executions turns very slow, or even the system cannot be able to solve the problem. For this reason, the Oblivion Rate is needed to remove pheromones in the system and alleviate the saturation of the system.



(a) Influence of the number of queens related to the number of pheromones deposited in the graph.

(b) Influence of the number of ants related to the number of pheromones created.

Figure 5.2: This figure shows the influence that the number of queens (Figure 5.2(a)), and the number of ants (Figure 5.2(b)), exert over the number of pheromones created in the system. Figure 5.2(a) shows the number of pheromones with a population of 25 ants when they are executed during 100 iterations. The problem represented in Figure 5.2(b) is composed by 25 queens with an increasing number of ants.

5.1.3 What type of Oblivion Rate works better?

For the *N-Queens* problem, two different types of Oblivion Rate functions have been analysed. The first one is an exponential negative function with a fixed exponent, the second function takes into account the saturation of the system measured using the number of pheromones created and the maximum number of pheromones that can be created.

Figure 5.3 shows the number of pheromones deposited in the graph when the Oblivion Rate is defined by Eq. 4.8, when the Oblivion Rate uses the Eq. 4.9, and when the system does not use the Oblivion Rate. In these experiments 100 ants try to solve a 25-Queens problem during 200 iterations. Each experiment is repeated 10 times, and Figure 5.3 shows the mean number of pheromones.

This figure shows how the Oblivion Rate reduces the number of pheromones in the system. The Oblivion Rate with the fixed exponent provides the highest reduction of pheromones. The problem with the Oblivion Rate that uses the saturation of the system is related to the total number of pheromones. This number is so big that the exponent is always close to 0. For example, in the case of 25 queens the total number of pheromones is 8.1×10^8, the maximum number of pheromones deposited in the system is 127481, thus the saturation of the system is 1.5738×10^{-4} and the number of pheromones removed is very low.

Queens	Rep.	Ants	Iterations	Result	Max Queens
25	20	25	100	22.85 ± 0.79	24
			200	23.45 ± 0.58	25
		50	100	23.25 ± 0.62	24
			200	23.85 ± 0.79	25
		100	100	23.45 ± 0.74	25
			200	24.05 ± 0.66	25
50	15	25	200	45.73 ± 0.85	47
			300	46.26 ± 1.06	48
		50	200	47 ± 1.09	49
			300	46.13 ± 1.08	48
		100	200	46.53 ± 1.02	48
			400	47.8 ± 0.75	49
75	10	200	300	71.5 ± 0.8	73
			500	71.3 ± 0.9	73
100	4	300	800	96.75 ± 0.83	98
150	2	400	900	146.5 ± 0.5	147
200	1	500	1000	194	194

Table 5.1: This table provides information about the number of queens, the iterations, the number of ants and the number of repetitions for each experiment. The column ”Result” shows the mean number of queens located in the chessboard and the standard deviation. Finally, the last column shows the maximum number of queens correctly placed in each set of experiments.

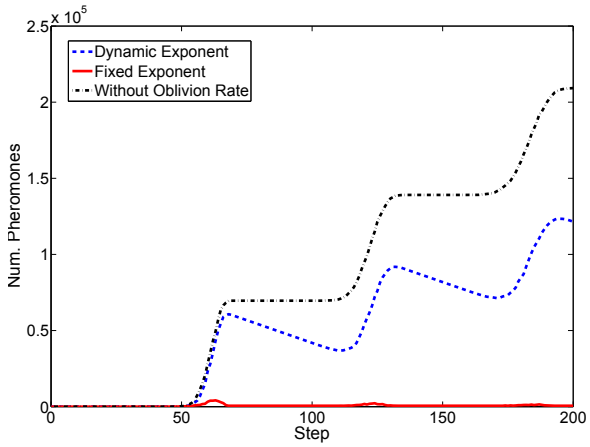


Figure 5.3: This figure shows the performance of the different Oblivion Rate functions. In this experiment, 100 ants try to solve the *25-Queens* problem in 200 steps.

5.1.4 Does this approach work?

Despite the advantages of the Oblivion Rate presented so far, the main question is whether this approach is able to solve the different problems, or not. Table 5.1 shows a summary of several experiments carried out in this work and the maximum number of queens placed in each experiment. All these experiments used the Oblivion rate with fixed exponent, because it is the Oblivion Rate that reduces more pheromones in the graph, and therefore allows the execution of bigger problems (such as *200-Queens* problem). There are some parameters that are not shown in Table 5.1 and which are meaningful for the execution of the algorithm. The values for these parameters are the same in all the executions:

- Evaporation rate: 5%.
- α: 1 and β:1.5.
- Pheromone update rate: 0.1.

This table shows how the proposed approach allows to obtain suboptimal solutions close to the complete ones. Therefore, although the proposed CSP model and the related ACO implementation shows promising results when it is applied to the *N-Queens* problem, its behaviour in other domains must be analyzed.

5.2 Resource-Constraint Project Scheduling Problem

In this section, the graph model shown in Section 4.1.2 has been applied to different problems belonging to *Resource-Constraint Project Scheduling Problem* (RCPSP). A complete description about RCPSP problems can be found in Section 2.2.1.2.

The problems analyzed in this thesis have been extracted from PSPLib[1] [Kolisch and Sprecher, 1996] because it offers a large set of problems with different characteristics, and also because the optimal solution for each problem is provided, or the best-known heuristic value [Kolisch and Hartmann, 2006].

Each problem in the PSPLib library is composed by a set of instances, and all the instances belonging to the same problem have the same characteristics in terms of number of jobs, number of modes, and number of renewable, and non renewable, resources.

In order to measure, or validate, the performance of the proposed model applied to RCPSP, different measures have been extracted from each analyzed problem. The first metric is the mean number of pheromones created in the system. This characteristic is one of the critical part of the model because the fast growth of the pheromones can produce problems with the performance of the system.

The second metric measures the efficiency of the model, measured as the quality of the found solutions. Each problem of the PSPLib library is composed by a set of instances, and the optimum (or best heuristic) value obtained so far by the research community is published. In order to measure the efficiency of the model, we measure the mean makespan obtained by the proposed model for each instance belonging to the same problem, and then, this mean makespan is compared against the mean makespan published by the research community for each instance of the same problem. With this metric, we can understand whether the pheromone control process (performed by the Oblivion Rate) affects to the quality of the found solutions.

5.2.1 The initial experiment

Due to the number of problems, instances and different characteristics that compose the PSPLib library, an initial study, using only some instances belonging to different problems, has been perform to analyze which are the characteristics that affect to the complexity of the problem. This library currently contains 12119 instances grouped in 24 different problems.

The characteristics of the different selected instances are shown in Table 5.2. This table shows the experiment identification, the number of jobs of the selected instance, the number of modes, and the number of renewable, and non renewable, resources.

The selected instances try to cover the whole problem parameters by studying how the number of jobs, the number of modes and the number of different resources affect to the solution of the problem. For this reason, experiments number 1, 2 and 3 corresponds to *Single-Mode* problems that differ in the number of jobs involved in each project. Experiments number 4 and 5 correspond to projects that differ in the number of different execution modes for each project. Finally, projects

[1] http://www.om-db.wi.tum.de/psplib/main.html

Experiment Id.	Jobs	Modes	Renewable Resources	Non Renewable Resources
1	30	1	4	0
2	60			
3	120			
4	16	1	2	2
5		5		
6	16	3	1	2
7			5	
8	16	3	2	1
9				3

Table 5.2: Description of the RCPSP used in this work. Each problem is defined by the number of jobs and the number of execution modes for each mode, and the number of Renewable, and Non Renewable, resources.

with different number of *renewable* resources are represented in experiments 6 and 7, whereas the experiments 8 and 9 differ in the number of *non-renewable* resources.

All these instances have been solved using the proposed model with a dynamic Oblivion Rate. Each experiment has been repeated 20 times, using a colony composed by 500 ants executed during 2000 steps. The evaporation rate for the phermones is fixed to 5%, the value of α is 1 and β is given the value 2.

For each experiment, we measure the minimum makespan obtained in the 20 repetitions, the mean makespan and its standard deviation. Then we compare these results against the minimum makespan obtained by the research community. The results are shown in Table 5.3 . Note that the minimium makespan marked with a star denotes best-known heuristic values, while those values that are not marked corresponds to the optimum value for the problem.

Experiments number 1, 2 and 3 corresponds to single-mode problems that differ in the number of activities that composed the project. With higher number of activities in the project, the resulting graph is more complex (i.e. it contains more constraints and more nodes). For this reason, the minimum makespan found by our approach get worst when the number of activities increases.

Another parameter that affects the performance of the algorithm is the number of different execution modes for each activity. When the problem is composed by activities with many execution modes, the minimum makespan is not really close to the optimum one. This is an expected results due to the number of different combinations in the pheromones.

Finally, regarding the number of different resources, we do not find any interesting difference between problems composed by many different types of resources and those projects with few different resources. Moreover, these experiments have found the optimal value, or a really close value to the optimal, for these instances.

Exp.Id	Best-known makespan	Proposed Approach	
		Min. makespan	Avg. makespan
1	65	**65**	69.9 ± 3.05
2	106*	110	130.44 ± 9.18
3	168*	195	239.46 ± 12.63
4	54	61	67.26 ± 4.66
5	32	39	66.73 ± 7.45
6	19	**19**	31.55 ± 3.87
7	25	**25**	39.06 ± 5.32
8	29	30	50.81 ± 5.52
9	32	33	39.88 ± 4.24

Table 5.3: Best-known makespan for each problem (PSPLIB), the minimum, the average and the standard deviation for the makespan obtained by our approach. Values marked by an asterisk represent the best-known heuristic values.

5.2.2 PSPLib in depth

In this section, a large set of problems belonging to PSPLib has been analyzed. The main characteristics of these problems are shown in Table 5.4. This table contains the name of the problem, the number of instances that compose the problem, the number of jobs that compose the project, the number of modes in which each job can be executed, and the number of renewable, and non renewable, resources.

All the instances contained in the problems listed in Table 5.4 have been solved using the proposed model without Oblivion Rate (called normal ACO), using a dynamic Oblivion Rate that takes into account the saturation of the system, and finally, using a CRO-based Oblivion Rate. Also, as ACO is a stochastic algorithm, each experiment has been repeated 10 different times.

The configuration of the ACO algorithm is the same in almost all the experiments carried out in this section. For all the problems, except *j90.sm* and *j120.sm*, the colony is composed by 100 ants that are executed during 400 steps. For the *j90.sm* and *j120.sm* problems, 200 ants have been used during 500 steps. The algorithm set-up, or configuration, has been changed because *j90.sm* and *j120.sm* are the most complex problems in RCPSP Library. In all these experiments the evaporation rate has been fixed to 5%, and α, and β take the values 1 and 2, respectively.

The first experiment solves all the instances for the problems using the proposed model with a dynamic Oblivion Rate, and without Oblivion Rate (Normal ACO). For each problem we have compared the maximum number of pheromones obtained in order to know whether the dynamic Oblivion Rate provides an important reduction, or not.

Table 5.5 shows the maximum number of pheromones created in the system using a dynamic Oblivion Rate, and without the Oblivion Rate.

Problem Name	Instances	Jobs	Modes	Renewable	Non Renewable
j30.sm	480	30	1	4	0
j60.sm	480	60			
j90.sm	480	90			
j120.sm	600	120			
j10.mm	536	10	3	2	2
j12.mm	547	12			
j14.mm	551	14			
j16.mm	550	16			
j18.mm	552	18			
j20.mm	554	20			
j30.mm	640	30			
m2.mm	481	16	2	2	2
m4.mm	555		4		
m5.mm	558		5		

Table 5.4: This table shows the main characteristics of the different problems analyzed in this thesis. For each problem, this table shows the number of instances that compose the problem, the number of jobs that compose the project, the number of modes, and the number of renewable, and non renewable, resources for each job.

As it can be observed in Table 5.5, the most complex problems are those belonging to Single Mode. This value is shown when no Oblivion Rate is applied. Nevertheless, the percentage reduction obtained in the system when the dynamic oblivion rate is used is really important. Note that for complex problems (Single-Mode problems) the minimum percentage reduction is around the 80%, but for Multi-Mode problems this percentage is not lower than 94%. This means that the Oblivion Rate reduces drastically the number of pheromones created in the system (as it happened in the *N-Queens Problem*).

Figure 5.4 shows the graphical evolution of the number of pheromones created in the system for *j90.sm* (Figure 5.4(a)) and *m5.mm* (Figure 5.4(b)). In this figure it can be observed the drastically reduction in the number of pheromones, in the case of *j90.sm* this reduction is 84.09%, and in the case of *m5.mm*, 97.85% . The rest of the experimental results for all the problems analyzed in this section can be consulted in Appendix A.

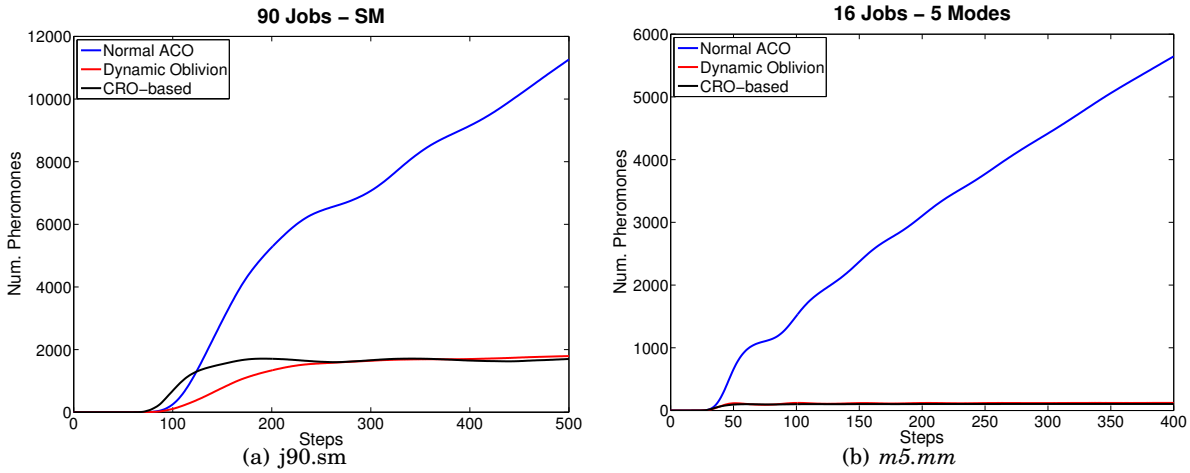
The next step is to analyze how this high percentage of reduction affects to the quality of the found solutions. In this case, we will compare (for each problem) the average minimum makespan of our approach with the minimum makespan published by the research community. Three different kinds of executions have been performed:

1. The proposed model without Oblivion Rate.
2. The proposed model using the dynamic Oblivion Rate.
3. The proposed model using a CRO-based Oblivion Rate.

In order to fix the size of the coral for the CRO-based Oblivion Rate, it has been used the maximum number of pheromones created in the system when the dynamic Oblivion Rate is used. In this way, we can compare the performance of the dynamic, and the CRO-based, Oblivion Rates.

Problem	Normal ACO	Dynamic Oblivion	Reduction pct.
j30.sm	1352	225	83.35%
j60.sm	3789	767	79.75%
j90.sm	11262	1791	84.09%
j120.sm	24781	4813	80.57%
j10.mm	2023	68	96.63%
j12.mm	2695	83	96.92%
j14.mm	3246	101	96.88%
j16.mm	3841	120	96.87%
j18.mm	4315	142	96.71%
j20.mm	4865	168	96.54%
j30.mm	5922	276	95.33%
m2.mm	2092	112	94.64%
m4.mm	4891	268	94.52%
m5.mm	5645	121	97.85%

Table 5.5: This table shows the maximum number of pheromones created in the system using a dynamic Oblivion Rate, and without the Oblivion Rate.



(a) j90.sm (b) *m5.mm*

Figure 5.4: This table shows the graphical evolution of the number of pheromones created in the system for *j90.sm* (Figure 5.4(a)) and *m5.mm* (Figure 5.4(b)), when the system is not using the Oblivion Rate, with the dynamic Oblivion Rate, and with the CRO-based Oblivion Rate.

Table 5.6 shows the maximum number of pheromones created in the system with the dynamic Oblivion Rate, the size of the coral reef in the CRO-based Oblivion Rate and the maximum number of attempts that a new coral (i.e. pheromone) has to find a place in the reef. Note that the reef is a squared two-dimensional matrix, for example, a reef size of 15 means that the reef has a dimension of 15×15.

Tables 5.7, 5.8, 5.9 show the average minimum makespan published by the research community, the average minimum makespan obtained by the proposed model: without using the Oblivion Rate (Normal), using a dynamic Oblivion Rate, and using a CRO-based Oblivion Rate. These tables also show the number of instances in which the ACO has reached the minimum makespan published by the research community and the percentage of instances where the minimum makespan obtained by the model is equal to the minimum makespan published (column "Efficiency"). Table 5.7 shows the results for the Multi-Mode problems with different number of modes. Table 5.8 shows the results for the Multi-Mode problems that differs in the number of jobs. And finally, Table 5.9 shows the results for the Single-Mode problems.

As it can be seen in these tables, the average minimum makespan obtained by our approach is really similar when the model is not using the Oblivion Rate, when the dynamic Oblivion Rate is used, and when the Oblivion Rate has a CRO-based behaviour. These results are really interesting, if we keep in mind the strong reduction percentage obtained when the Oblivion Rate is used. The Oblivion Rate (dynamic or CRO-based) reduces at least the 80% in the Single-Mode problems, and 94% for Multi-Mode problems, obtaining solutions really close to the ones obtained by the system without controlling the number of pheromones, and thus having more information about the past of the algorithm. Finally, all these makespan values obtained by our approach are really close to the optimal makespan obtained by the research community. For previous reasons, we can conclude that the Oblivion Rate is a good metaheuristic for RCPSP problems because the system obtains solutions close to the optimal, and the number of pheromones created in the system has been extremely reduced.

Problem	Max. Pheromone Dynamic Oblivion	Reef Size.	Attempts
j30.sm	225	15	50
j60.sm	767	27	100
j90.sm	1791	42	100
j120.sm	4813	69	100
j10.mm	68	9	10
j12.mm	83	10	10
j14.mm	101	10	10
j16.mm	120	12	10
j18.mm	142	12	10
j20.mm	168	13	10
j30.mm	276	17	10
m2.mm	112	11	10
m4.mm	268	17	10
m5.mm	121	12	10

Table 5.6: This table shows the coral reef size for the CRO-based Oblivion Rate, taking into account the maximum number of pheromones created in the system using the dynamic Oblivion Rate. Also the number of attempts that a new coral (i.e. a pheromone) has to find a place in the reef is shown.

Dataset	MinPubl.	Oblivion	Obt.Min	Coincidences	Efficien
m2.mm	30.16 ± 6.87	Normal	31.04 ± 7.42	245	50.94%
		Dynamic	31.04 ± 7.4	245	50.94%
		CRO	31.03 ± 7.33	252	52.39%
m4.mm	22.71 ± 7.3	Normal	26.69 ± 8.61	11	1.98%
		Dynamic	26.68 ± 8.51	13	2.34%
		CRO	26.58 ± 8.42	19	3.42%
m5.mm	21.16 ± 8.14	Normal	25.73 ± 9.4	2	0.36%
		Dynamic	25.76 ± 9.31	1	0.18%
		CRO	25.95 ± 9.35	0	0.0%

Table 5.7: This table shows the average minimum makespan published by the research community, the average minimum makespan obtained by the proposed model without using the Oblivion Rate (Normal), using a dynamic Oblivion Rate, and using a CRO-based Oblivion Rate, and the efficiency for each approach. In this table, the problems analyzed differs in the number of execution modes for each task.

Dataset	MinPubl.	Oblivion	Obt.Min	Coincidences	Efficien
j10.mm	19.04 ± 6.21	Normal	19.69 ± 6.46	309	57.65%
		Dynamic	19.68 ± 6.46	311	58.02%
		CRO	19.66 ± 6.42	307	57.28%
j12.mm	21.34 ± 6.48	Normal	22.36 ± 6.63	219	40.04%
		Dynamic	22.25 ± 6.54	217	39.67%
		CRO	22.38 ± 6.69	213	38.94%
j14.mm	23.18 ± 6.14	Normal	25.23 ± 6.73	114	20.69%
		Dynamic	25.31 ± 6.79	107	19.42%
		CRO	25.18 ± 6.70	117	21.23%
j16.mm	24.93 ± 6.02	Normal	27.67 ± 7.03	77	14.0%
		Dynamic	27.76 ± 7.07	74	13.45%
		CRO	27.71 ± 6.95	66	12.0%
j18.mm	26.57 ± 6.47	Normal	29.99 ± 7.66	41	7.43%
		Dynamic	30.09 ± 7.54	56	10.14%
		CRO	29.97 ± 7.57	57	10.33%
j20.mm	27.71 ± 6.99	Normal	32.08 ± 8.62	25	4.51%
		Dynamic	32.08 ± 8.76	30	5.42%
		CRO	32.05 ± 8.86	25	4.51%
j30.mm	28.79 ± 7.44	Normal	36.16 ± 17.39	2	0.31%
		Dynamic	36.16 ± 17.44	7	1.09%
		CRO	36.14 ± 17.41	4	0.62%

Table 5.8: This table shows the average minimum makespan published by the research community, the average minimum makespan obtained by the proposed model without using the Oblivion Rate (Normal), using a dynamic Oblivion Rate, and using a CRO-based Oblivion Rate, and the efficiency of each approach, using the Multi-Mode problems that differs in the number of jobs that compose the project.

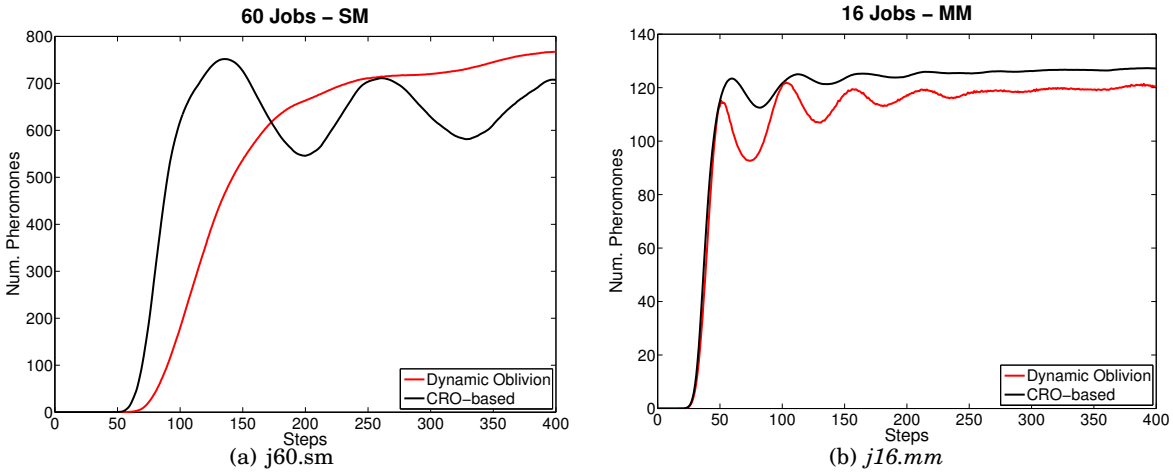
Dataset	MinPubl.	Oblivion	Obt.Min	Coincidences	Efficien
j30.sm	58.99 ± 14.07	Normal	59.98 ± 14.75	280	58.33%
		Dynamic	60.0 ± 14.78	280	58.33%
		CRO	60.0 ± 14.77	279	58.13%
j60.sm	79.8 ± 17.42	Normal	82.78 ± 20.68	243	50.62%
		Dynamic	82.83 ± 20.73	241	50.21%
		CRO	82.65 ± 20.57	250	52.08%
j90.sm	94.94 ± 20.57	Normal	99.1 ± 25.62	250	52.08%
		Dynamic	99.07 ± 25.6	251	52.29%
		CRO	99.11 ± 25.66	249	51.88%
j120.sm	122.19 ± 39.75	Normal	135.73 ± 49.18	48	8.0%
		Dynamic	135.76 ± 49.24	48	8.0%
		CRO	135.72 ± 49.09	51	8.5%

Table 5.9: In this table the results for the Single-Mode problems are shown. This table contains the average minimum makespan published by the research community, the average minimum makespan obtained by the proposed model without using the Oblivion Rate (Normal), using a dynamic Oblivion Rate, and using a CRO-based Oblivion Rate, and the efficiency of each approach.

Analyzing the standard deviation of the mean makespan, it can be seen how when the size of the problem, and thus its complexity, increases, the standard deviation increases too. As it may seem a bad result, the standard deviations obtained by the model using *Oblivion Rate* is really similar to the standard deviation obtained without using this metaheuristic. For this reason, we can conclude that the *Oblivion Rate* does not affect to the standard deviation of the makespan.

The efficiency of the algorithm is computed as the number of instances where the proposed model is able to obtain the minimum solutions published by the research community, divided by the total number of instances in the problem. Analyzing the performance of the proposed system using this efficiency metric, it can be seen that for the Multi-Mode problems that differ mainly in the number of jobs (Table 5.8), the dynamic Oblivion Rate works better than the CRO-based Oblivion Rate, because the dynamic Oblivion Rate shows higher efficiency in 5 out of 7 problems. On the other hand, regarding the Multi-Mode problems that differs mainly in the number of modes (Table 5.7), CRO-based Oblivion Rate is better than dynamic Oblivion Rate. Finally, analyzing the results from Single-Mode problems, the performance of both approaches are really similar because both approaches show the better efficiency in the 50% of the problems, i.e. dynamic Oblivion Rate has higher efficiency in 2 out of 4 problems.

Regarding to the number of pheromones created in the system by the dynamic and the CRO-based Oblivion Rate, the evolution is really similar. For the Single-Mode problems, the number of pheromones with the CRO-based Oblivion Rate suffers more variations due to the fight for the space performed (Fig. 5.5(a)). For the Multi-Mode problems, the number of pheromones, in both approaches, grows in the first steps of the algorithm, then it starts to oscillate, until it finally keeps an stable value (Fig. 5.5(b)).



(a) j60.sm (b) *j16.mm*

Figure 5.5: This table shows the graphical evolution for the number of pheromones created in the system in *j60.sm* (Figure 5.5(a)) and *j16.mm* (Figure 5.5(b)) problems, when the system is using the dynamic Oblivion Rate, and the CRO-based Oblivion Rate. For the Single-Mode problems, the number of pheromones with the CRO-based Oblivion Rate suffers more variations due to the fight for the space performed. For the Multi-Mode problems, the number of pheromones, in both approaches, grows in the first steps of the algorithm, then it starts to oscillate, until it finally keeps stable values.

5.3 The Lemmings video game

In this section, the proposed model has been applied to *The Lemmings* video game (described in Section 2.2.2.2). The goal of this section is to validate whether the proposed model can be applied for solving levels of *The Lemmings* game, and to measure the performance of this model.

In this work 14 different levels have been designed and implemented (see Appendix B for a further description of those levels). These levels have been classified in three categories: easy, medium and hard, depending on the complexity, or difficulty, of the level. This complexity is determined (by hand) taking into account the size of the level, the different blocks contained in each level, the distance from the entry point to the exit point, the number of skills needed to solve the level, and finally the type of terrains contained in the levels.

Figure 5.6 shows a hard level used in this thesis. As it can be seen in this figure, the size of the level is big, and all the terrains are editable. This introduces a higher complexity because there are a lot of possible paths that could kills the lemmings.

Two different experiments have been carried out using previous levels. The goal of the first one is to compare GA and ACO algorithms when trying to solve the levels of The Lemmings game, this comparison has been performed by taking into account the number of different successful paths that each algorithm is able to find. A successful path is the one that connect the entry point with the exit point of the level.

The second experiment analyzes the influence of the *granularity* of an ACO algorithm. The granularity is defined as those possible representations which can be used, in this kind of domains, to manage the set of ants that will try to solve the level. In this case, three different

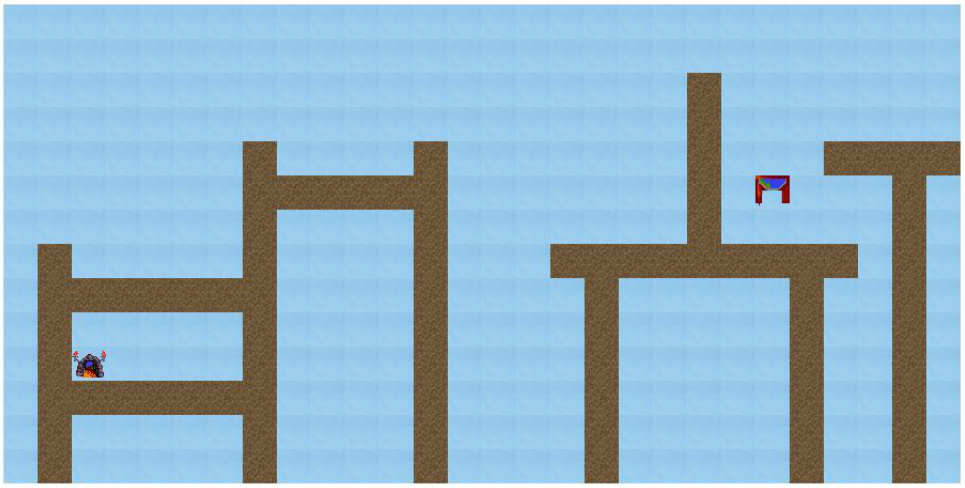


Figure 5.6: This figure shows an example of a hard level (Level 12, see Appendix B).

representations level are used. For these representations one simple ant will represent from a single lemming to a whole population of lemmings.

5.3.1 Comparing GA vs. ACO

As it has been previously explained, this first experiment compares the performance of a GA and the proposed model for solving levels of *The Lemmings* video game.

In this set of experiments, we have modelled a GA, and two ACO behaviours for the proposed model. The first ACO behaviour is called *Random Ant*, and the second one is called *Common-sense*. Next subsections provide the details of previous algorithms.

5.3.1.1 Genetic Algorithm approach

The GA applied in this work, initializes individuals with a random genotype length. The maximum length of the genotype depends on the maximum time of the level or the maximum genotype length allowed. This genotype is a list of genes where each gene ($\langle T, S \rangle$) contains the skill (S) that is going to be executed in the step T. Both values (the step and the skill) are selected randomly depending on the maximum time given to solve the level, and the total number of remaining skills. The genotype represents the different decisions that the player could make. This genotype is then evaluated against the level. The lemming starts its execution applying the skills specified in the given steps.

The goal is to maximize the fitness function represented in Section 4.2.2. This fitness function takes into account the number of lemmings that have reach the exit, the number of skills applied, and the time needed to solve the level.

Although the proposed ACO model will use the same function to evaluate the goodness of the paths, only GA can produce negative fitness value. This negative value is obtained when the individual produces an invalid path (i.e. in the evaluation, the lemming is not able to reach the exit point or the lemming dies trying it). In this case, the fitness value is $F(Ind_i) - 1$, and this value makes paths really close to the optimal solution are evaluated with fitness value close to 0, whereas the corresponding values for the worst paths will be close to -1.

5.3.1.2 Ant Colony behaviours

Regarding the ants behaviour, two different behaviours have been implemented in this experiments: *Random Heuristic* and *Common-Sense Heuristic*.

To select the next movement by *Random Heuristic* ants, they will have two options. The first one consists on executing the kill used in previous steps, the second one will be to select randomly a new skill from the set of available skills. The only restriction in the selection of a new skill is that any skill can be selected only if it can be assigned at least once.

The second behaviour, called *Common-Sense Heuristic*, filters the possible skills to apply taking into account the type of terrain of the surrounded nodes where the ant is placed. This behaviour avoid the execution of non-real movement such as a lemming is falling and in the air, it starts building a bridge. Therefore, this kind of assignment will be a context-based selection (from an agent-based perspective this context-based decision can be seen as an information diffusion problem where a set of agents, or "ants", must to decide what kind of information is really useful and how to apply it in their environment [Gonzalez-Pardo et al., 2012].

Once the ants have finished their execution, they go back to the nest (entry point of the level) and they update the pheromone values with the evaluation of the follow path using the equations described in Section 4.2.2.

5.3.1.3 Experimental setup and experimental results

All the levels have been solved using GA, ACO with random heuristic, and ACO with the common-sense heuristic. Each experiment has been repeated 50 different times, and the number of different found paths are used to compare the performance of these algorithms.

The configuration of the different parameters used in the GA are shown in Table 5.10 whereas Table 5.11 shows the configuration for the ACO experiments.

The main question of this experimental phase could be: *How many different successful paths can these algorithms find?*. This is a question about how many different solutions each algorithm is able to find. This information has been showed in Table 5.12. The values correspond to the number of different paths found in 50 executions of the algorithms.

Dealing with easy levels, both algorithms found a large set of different paths. In general, Genetic Algorithms and ACO with the random heuristic finds more paths than ACO with the common-sense heuristic. This is produced because GA and the random ACO are able to explore different actions that allows them to find more solutions.

Parameter	Value
Population Size	100
Max. Genotype length	20
Generations	500
# Offsprings	1
Crossover Rate	90%
Crossover Type	One point
Mutation Rate	1%
Elitism	No

Table 5.10: Configuration of the experiments with GA to solve the designed *Lemmings* levels.

Parameter	Value
# Ants	100
# Steps	500
Evaporation Rate	1%
α	1
β	1

Table 5.11: This table describes the configuration for the ACO experiments in the Lemmings domain. The parameters showed in this table are: the number of ants that compose the colony, the number of the simulation steps, the values for α and β, and the evaporation rate that is used in the *stigmergy* process.

Level	Complexity	Genetic Algorithm	ACO (Random Ant)	ACO (Common-Sense)
1	Easy	3219	3868	2516
2	Easy	12629	4463	4042
3	Easy	2364	446	1330
4	Easy	370	1130	2487
5	Medium	162	575	2520
6	Medium	649	2128	2560
7	Medium	1	12	228
8	Medium	7	348	326
9	Medium	2	18	62
10	Hard	54	35	**157**
11	Hard	0	3	**7**
12	Hard	2	15	**32**
13	Hard	164	26	35
14	Hard	1	4	**23**

Table 5.12: Number of different solutions found by GA and ACO algorithms.

Taking into account medium and hard levels, ACO (with both heuristics) finds more solutions than GA. This is an expected result because ACO gradually builds the solutions and GA needs to create the whole path from the beginning. When the size of the level increases, the difficulty to build a valid solution from the beginning increases,and even it is not possible to find any valid path (solution) in hard levels.

Finally, when the problem is medium or hard, ACO with the common-sense heuristic finds more paths than ACO with the random heuristic. This is produced because the common-sense heuristic uses the environment of the ant to filter the set of possible actions to execute. On the contrary, the random heuristic makes ants to explore solutions that produce the death of the lemming.

5.3.2 Evaluating the representation granularity of the ACO algorithms

The main conclusion from the previous experimental phase is that ACO algorithm performs better than GA to find the paths (solutions) that reaches the exit of the levels. Nevertheless, regarding any ACO algorithm is important to understand the effects that the representation of the ants exert in the solutions found by the algorithm.

In this set of experiments, two different kinds of simulations have been carried out to analyse the ants behaviour, and the contextual-based information exchange between them, for the ACO algorithm [Gonzalez-Pardo et al., 2012, Gonzalez-Pardo et al., 2014b]. On the one hand, a *micro simulation*, where each ant is used to model a Lemming, and a *macro simulation* where a complete swarm of Lemmings is represented using only one ant. The main characteristics of both simulations can be summarized as follows:

1. In the *micro simulations* two different approaches have been used: "one to one sequential" ($1to1_S$) and "one to one parallel" ($1to1_P$). In the $1to1_S$ simulations, only one action (the application of one skill) from each Lemming is allowed per step. Therefore, any Lemming must take into account the modifications than the rest of the Lemmings previously have made in the environment, so this kind of simulation can be considered as contextual-based, because the actions previously made by others Lemmings will affect to the current (scheduled) Lemming decision. In the $1to1_P$ simulations, in each step all of the available Lemmings can make one action ignoring the contextual information from the environment. The main difference between both kind of approaches is related to the contextual graph modification, the first one will provide a smooth modification of the graph, increasing the importance of the common-sense heuristic. The second approach will allow a fast modification of the graph, so the relevance of the meta-heuristic will be lower in the solution process.

2. In the *macro simulations*, denoted as $1toN$ (one to N), only the first Lemming from the swarm can execute an action (skill) in a particular step, whereas the rest of the Lemmings must follow the "leader" [Gonzalez-Pardo and Camacho, 2013a]. This kind of simulations provides a semi-static modification of the contextual graph. The graph is slowly modified, so the relevance of the meta-heuristic and the pheromone values will be increased. The parallel simulation is similar to the $1to1_S$ ones, but the latter allows to explore better the solution space (any Lemming has an opportunity to apply an skill), whereas the first reduce the solution space by following a particular Lemming leader.

Simulation	Graph modification	Contextual inf.	Num. Pheromones
$1to1_S$	medium	high	high
$1to1_P$	high	low	low
$1toN$	low	very high	very high

Table 5.13: Lemmings simulations and their related basic features.

Level	Complexity	$1to1_S$	$1to1_P$	$1toN$
1	Easy	**0,86 ± 0,009**	0,69 ± 0,084	0,65 ± 0,089
2	Easy	**0,88 ± 0,021**	0,85 ± 0,032	0,83 ± 0,040
3	Easy	**0,94 ± 0,038**	0,94 ± 0,012	0,90 ± 0,024
4	Easy	**0,96 ± 0,015**	0,95 ± 0,008	0,94 ± 0,019
5	Medium	**0,91 ± 0,005**	0,88 ± 0,031	0,85 ± 0,040
6	Medium	**0,94 ± 0,000**	0,89 ± 0,027	0,86 ± 0,037
7	Medium	0 ± 0,000	0,90 ± 0,020	**0,91 ± 0,044**
8	Medium	0 ± 0,000	**0,93 ± 0,006**	0,63 ± 0,018
9	Medium	0,69 ± 0,034	0,69 ± 0,093	**0,78 ± 0,110**
10	Hard	0 ± 0,000	0 ± 0,000	**0,67 ± 0,052**
11	Hard	0 ± 0,000	0 ± 0,000	**0,88 ± 0,050**
12	Hard	0,72 ± 0,061	**0,91 ± 0,013**	0,75 ± 0,034
13	Hard	0,78 ± 0,002	0,82 ± 0,029	**0,90 ± 0,052**
14	Hard	0 ± 0,000	**0,95 ± 0,003**	0,94 ± 0,020

Table 5.14: Average and standard deviation of the best solutions found by the ACO algorithm under different simulation configurations. These results have been obtained executing the different algorithms 50 different times.

Previous simulations allows to analyse the behaviour of our approach by modifying three essential features: how fast the graph could be modified, how affects the contextual information to the searching process, and finally the importance of the pheromone concentration in the searching process. Table 5.13 shows a summary of both, the simulations designed and their basic features (using a qualitative estimation).

All the experiments have been repeated 50 times, using the common-sense heuristic behaviour for the ants. In each experiment, the ant colony is composed by 100 ants that execute during 500 steps. The evaporation rate of the system is 1%, and α and β parameters (needed to measure the influence of the heuristic and the pheromone values) are fixed to 1. The number, and quality of the different found paths (solutions), have been used to compare the performance of our approach.

Table 5.14 and Figure 5.7, shows the results of our approach with the three different simulations carried out. Figure 5.7 shows the number of different paths (solutions) found by each algorithm, whereas Table 5.14 shows the average and standard deviation of the solutions quality.

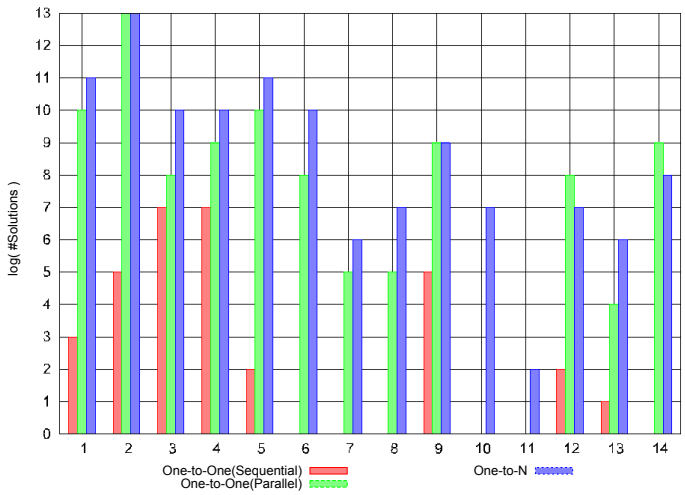


Figure 5.7: The figure shows the number of different solutions found by the algorithms. The Y axis represent the $\log_2$ of the different solutions found by the algorithms.

Analyzing the quality of the solutions (Table 5.14) the $1to1_S$ approach obtains better solutions that $1to1_P$ and $1toN$ in easy and medium levels. This is produced by two different reasons. On the one hand, with easy and medium levels the solution approach is not as bigger as the one in hard levels. So a sequential depth-first search is able to find good solutions, in the maximum number of simulation steps allowed. In this kind of levels is better to strongly use the contextual information than to make a wide parallel search. On the other hand, $1to1_S$ approach performs a depth-first search in the solution space, while $1to1_P$ and $1toN$ algorithms make a breadth-first search.

The Figure 5.7 shows the number of different paths that each algorithm is able to identify. In general, $1toN$ and $1to1_P$ finds more paths than $1to1_S$. This effect is produced because the solution space is more explored by the $1toN$ and $1to1_P$ than $1to1_S$. In a single execution of $1to1_P$ and $1toN$, the number of parallel searches are equals to the number of ants that compose the colony, while in $1to1_S$ al the ants compose a single search. Comparing $1to1_P$ and $1toN$ can be seen that $1toN$, in general, finds more different paths that $1to1_P$. This is an expected results because although both algorithms make a breadth-first search, $1toN$ makes more parallel searches than $1to1_P$.

Once these different experiments have been carried out, one issue, related to the amount of pheromones stored in the graph, was detected. Even for the hardest levels and using the $1toN$ approach, the simulation never creates a significant number of pheromones that could overload the system. Therefore, it was considered that using these levels no pheromone control metaheuristics (Oblivion Rate and CRO-based) should be applied.

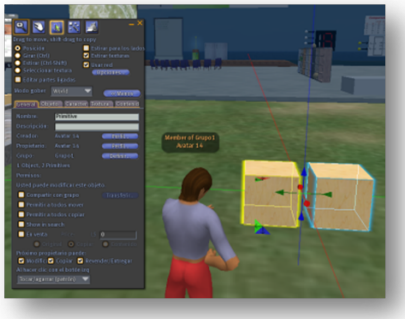


Figure 5.8: The figure shows an avatar building some objects in VirtUAM.

5.4 Clustering Behaviours in Virtual Worlds

This section provides a description about the virtual environment where the experiment takes place (VirtUAM), the description about the avatar information that has been used in this thesis, and finally, the empirical results obtained.

5.4.1 Virtual World Environment

In this work, the VW is composed by only two islands, i.e. two VW regions. The first one (called *Main island*) is used as an example island containing different objects that can be used by the different users for training. Any object can be analysed taking into account its **3D details**, that is the visual aspect of the object, and its **programmed behaviour** using LSL scripts (LSL stands from *Linden Scripting Language*). This island is used to guide users by providing them different examples.

The second island, called *Sandbox island*, has been designed to place the different experiments. This island contains several parcels and the users must select one of those parcels to build there whatever they want. Figure 5.8 shows an avatar building in its parcel. There are three different types of structures that the avatar could decide to build:

1. A composition focused mainly in the graphical design. This composition is built using a set of objects and the 3D aspect can be very detailed. Most of the effort carried out is dedicated to the graphical aspect so little programmed functionality is expected.

2. A composition with simple graphical aspect but with a complex programmed functionality. This case is the opposite to the previous one, in this composition the graphical aspect is not really important because most of the effort is applied to program the functionality with scripting files.

3. A mixed between previous cases, where the composition is not very realistic but it has some programmed functionality.

The selected users (56 students from last high school courses, during 2 years) decide what they are going to build and which is the proportion between 3D details and programmed behaviour. With this goal, different student behaviours are expected to appear in the VW. For example, the behaviour (in terms of actions carried out inside the virtual world) of those students focused on programming should differ from the behaviour of those students building a realistic composition.

5.4.2 Analyzing the avatar position

From all of the available information in the VW (see Section 2.2.3), this work uses mainly the **Avatar position** in form of GPS coordinates. This information is automatically gathered every 5 seconds, only if the current position is different to the last registered position for this avatar. This check avoids the insertion of duplicated information in the database. The avatar position can be used to define communities composed by those students working in near places. The algorithm used to create avatar communities based on the avatar position was *K-means* [MacKay, 2003, Macqueen, 1967].

K-means is a popular and well known partitional clustering algorithm. It is a straightforward clustering guided method (usually by a heuristic or directly by a human) to classify data in a predefined number of clusters. Given a fixed number of clusters, K-means tries to find a division of the dataset based on a set of common features given by distances or metrics that are used to determine what elements belong to each cluster.

K-means algorithm depends on a parameter called k. This parameter fixes the number of clusters in which data are going to be grouped. There are different ways to specify the value for this parameter:

1. **Giving a range of values**. This approach tries to group data using different values of k between a *min* and *max* values that must be specified.

2. **Giving a value determined by the problem**. In this case, the modelled problem determines the fixed number of clusters.

3. **Determining the value using statistical measures**. Some other works use the Bayesian information criterion [Ishioka, 2000, Pelleg and Moore, 2000] over data generated by Gaussian distributions. Other works assume that data are generated by a Poisson distribution [Hardy, 1996] and other papers work with Monte Carlo techniques [Halkidi et al., 2002]. In all these cases the statistical approach used generates the best value for k.

4. **Giving the same value as the number of classes**. This option is used with classification techniques and it is very similar to option 2. In this case, the user wants to classify data into a given number of classes. Therefore the number of clusters is the same as the number of classes, because each class is represented by one cluster.

5. **Specifying the value using data visualization**. This technique uses the data visualization to determine the number of clusters. This option can be considered as *Supervised* clustering. The disadvantage of using this approach is that sometimes it is not possible to use this method because there is so many information that the visual identification of the number of clusters is very difficult.

In this work, the algorithm described in [Pham et al., 2005] has been implemented and used. Authors use a function to evaluate the goodness of a given value of k, this function ($f(K)$) takes into account the distance between each value to its corresponding centroid penalizing those clusters composed by only one data. In order to define the value of k, authors execute the *k-means* several times changing the value of k, from 1 to N clusters, where N represents the number of elements that compose the dataset. For each k-value, the value $f(k)$ is computed. Finally, the k-value with lowest $f(k)$ is considered as "the best" value for k.

Although K-means algorithm has been used in many different domains, and it has been improved several times [Kanungo et al., 2002, Lai and Liaw, 2008, Wu and Yu, 2009] or even combining k-means with other algorithms such as Genetic algorithms [Min and Siqing, 2010, Prabha and Saranya, 2011], in this work the classical algorithm has been used. Applying the K-means algorithm to cluster avatar position, the communities composed by students working in near places will appear. This information is useful because in VWs students are free to move around the world and thus students would go away from the teacher and not pay attention to the given task. The main goal from this thesis work is to compare this straightforward application of K-means against a new approach based on ACO algorithms applied to clustering problems [Gonzalez-Pardo and Camacho, 2014a].

5.4.3 Analyzing the avatar behaviour with the *Normalized Compression Distance*

Also, we can analyze the avatar behaviour to group together those users that perform similar actions inside the VW. TO do that, we have analyzed the different behavioural files created from the interactions of the avatars. In order to classify the avatar behaviour a metric is needed, but this metric should takes into account that the behavioural model is based on a text representation. For this reason, the *Normalized Compression Distance (NCD)* has been selected. This metric provides a measure of the similarity between two objects, x and y, using compressors. The definition is as follows:

$$NCD(x,y) = \frac{\max\{C(xy) - C(x), C(yx) - C(y)\}}{\max\{C(x), C(y)\}}, \tag{5.1}$$

where C is a compression algorithm, $C(x)$ is the size of the C-compressed version of x, and $C(xy)$ is the compressed size of the concatenation of x and y. NCD generates a non-negative number $0 \leq NCD(x,y) \leq 1$. Distances near 0 indicate similarity between objects, while distances near 1 reveal dissimilarity.

The NCD is just one of the many similarity distances that use compression algorithms. Other distances [Benedetto et al., 2002, Kraskov et al., 2005, Zhang et al., 2007] are small variations and can be easily reduced to it, as it is possible to prove that this distance is as good as any other that can be computed by a universal Turing machine [Turing, 1936]. However, this metric has been successfully used in a wide number of domains based on text mining and classification [Gonzalez-Pardo et al., 2010c, Helmer, 2007, Telles et al., 2007].

CompLearn Toolkit [Cilibrasi et al.,] is used to compute and visualize the resulting NCD-based clustering described in [Cilibrasi and Vitanyi, 2005]. This clustering algorithm comprises two phases. First, the NCD matrix is calculated using a compression algorithm, in this case LZMAX, which is a Lempel-Ziv-Markov chain algorithm [Pavlov, 1998]. Second, the NCD matrix is used as input to the clustering phase and a dendrogram is generated as output. A dendrogram is an undirected binary tree diagram, frequently used for hierarchical clustering, that illustrates the arrangement of the clusters produced by a clustering algorithm.

5.4.4 Experimental Results

The data used in this section corresponds to the tracking activities of 7 non-skilled users working in the VW during a whole week (4 hours per day), during 4 weeks (28 users) and this experiments where carried out in two different years (56 users).

5.4.4.1 Using *K-means* algorithm for clustering students taking into account the eye-gaze information

This experiment involves 8 avatars (7 students and a teacher). The teacher was in charge of explaining some concepts about Ancient Greece while students must pay attention to the lecture. The goal of this experiment is to determine whether a particular avatar is paying attention or not to the lesson. To do that, only the avatar position and eye-gaze data are analyzed using the K-means algorithm. This algorithm will divide the set of recorded data into two classes corresponding to the students that are paying attention and those who are not.

Figures 5.9 and 5.10 represent information about avatars. Each asterisk represents an avatar in a specific time. The information specified for each avatar includes the distance from the avatar to the lecturer and the angle formed by the user-camera view and the distance just explained (eye-gaze). For example, in Figure 5.9, the circled asterisk represents an avatar that has a distance of 9 meters to the lecturer and an eye-gaze angle of around 100 degrees.

Figure 5.9 shows distance from the avatars to the lecturer and the avatar's eye-gaze before the lecturer requires their attention, and Figure 5.10 represents the same data but when the lecture has already started. As it can be seen, values for distance and eye-gaze when the lecture has started are more compacted than in Figure 5.9. This is an expected result because when the teacher is giving explanations, students are, more or less, within the same distance to the teacher.

Figure 5.10 shows an example of the result of K-means over the data taken during the lecture. It can be seen that the variable used to perform the clustering is *eye-gaze* which means that the task of determining whether an avatar is paying attention to the teacher depends exclusively on the gaze of the user and not on the distance from the avatar to the teacher. Results show that the angle that describes the behaviour of the user is 36.45 degrees. This means, that if the gaze of a user is greater than this value, the user is not paying attention to the explanation given by the teacher. Nevertheless, this result is extremely dependent on the dataset used in the experiments. In order to determine the information that defines whether a student is paying attention or not, a more complete study should be performed.

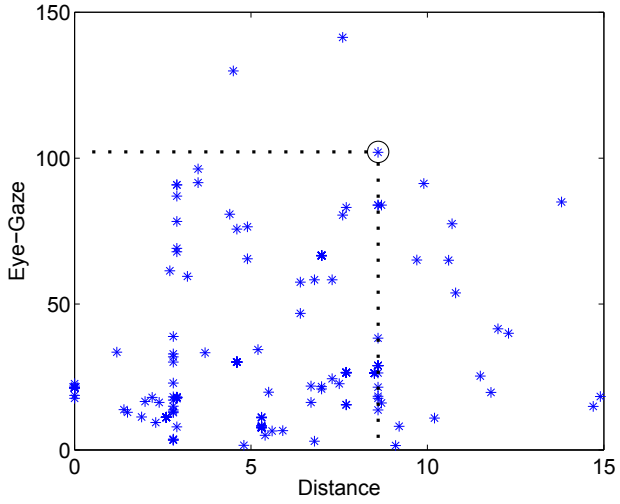


Figure 5.9: Representation of the avatars angle of vision (y-axe) and their distance to the teacher (x-axe), before the lecturer requires their attention.

5.4.4.2 Clustering avatar behaviour using the *NCD*

In this second experiment, two teachers interact with the different students inside the VW. The goal of this experiment is to determine whether the behaviour of the avatars can be used to group together those avatars with similar behaviour.

Analysing the different files that contain the behaviour of the different avatars, a new problem related to the size of the resulting files appears. The larger the student session, the bigger resulting files. This unfortunately difficult the NCD analysis because it needs huge computational resources. For this reason, some new representation schemas are needed to simplify and reduce the size of these files.

Four different representations was studied and analysed:

1. **Representation #1**. This representation is the one showed in Figure 2.15, where the actions performed are stored in chronological order.

2. **Representation #2**. In this representation, actions are simplified with a number indicating the number of consecutive repetitions of this action. For example, "*2MF3BT1XA*" written using representation #1 means *MFMFBTBTBTXA* (i.e. 2 times *Move Flying*; 3, *Build Touching* and 1 *Chat All*).This representation reduce the size of the behavioural files, as other compression techniques used to reduce the size of the file deleting repeated information.

3. **Representation #3**. In this type of representation, the number following each action represents the total number of times that this action has been reproduced. With this representation, "*2MF3BT1XA*" does not means the same as representation #2, but it means that avatar has moved flying (MF) 2 times (in the whole session). In this case the previous example could be "*MFBTMFXABTBT*", or "*MFBTXABTMFBT*" or "*MFBTBTMFBTXA*. This representation keeps the appearance order, this means that two behavioural files

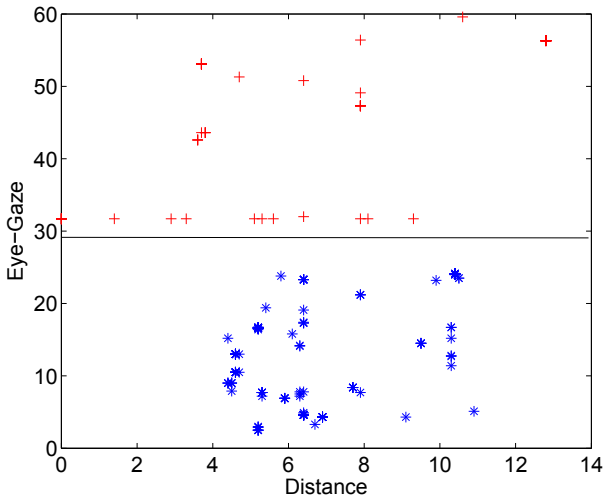


Figure 5.10: Result of the K-means algorithm applied to avatars angle of vision (y-axe) and their distance to the teacher (x-axe) during the first stage of the lecture.

could not start with the same sequence of actions, because one avatar can start the session building new objects (MF) while other avatar start going for a walk inside the VW.

4. **Representation #4**. Finally, this representation is the same as representation #3 but it keeps the same order all the time. This means that behavioural files have the following structure:

$$\alpha\mathbf{XA}\beta\mathbf{MF}\gamma\mathbf{MR}\delta\mathbf{BT}\epsilon\mathbf{BC}\rho\mathbf{XO}\eta\mathbf{MP} \tag{5.2}$$

Where $\alpha,\beta,\gamma,\delta,\epsilon,\rho,\eta$ are the number of times that the actions are repeated. Using this representation the size of the behavioural files are strongly reduced but this representation lacks of temporal representation.

Finally, the results of the NCD applied to the behavioural files with these representations are shown in Figure 5.11. This figure shows the result of applying *NCD* over the behavioural files using the described representations. Representation #1 is show in subfigure a, whereas representation #2 is shown in subfigure b. And subfigures c and d show representations #3 and #4 respectively.

There is a trade-off between the size of the behavioural files and the accuracy of the NCD. On one hand, representation #1 is the more detailed representation (and thus it provides the more precise clustering results) but the resulting behavioural files can be extremely large complicating the *NCD* clustering task. On the other hand, representation #4 is the most compacted representation but *NCD* results are not very precise. This is shown in Figure 5.11, where avatar with similar behaviours (i.e. the communities composed by *Avatar1* and *Avatar4*, and *Teacher1* and *Teacher2*) are very close using representation #1 (Subfigure a) but if the representation is less precise, these avatars appear more separated in the corresponding

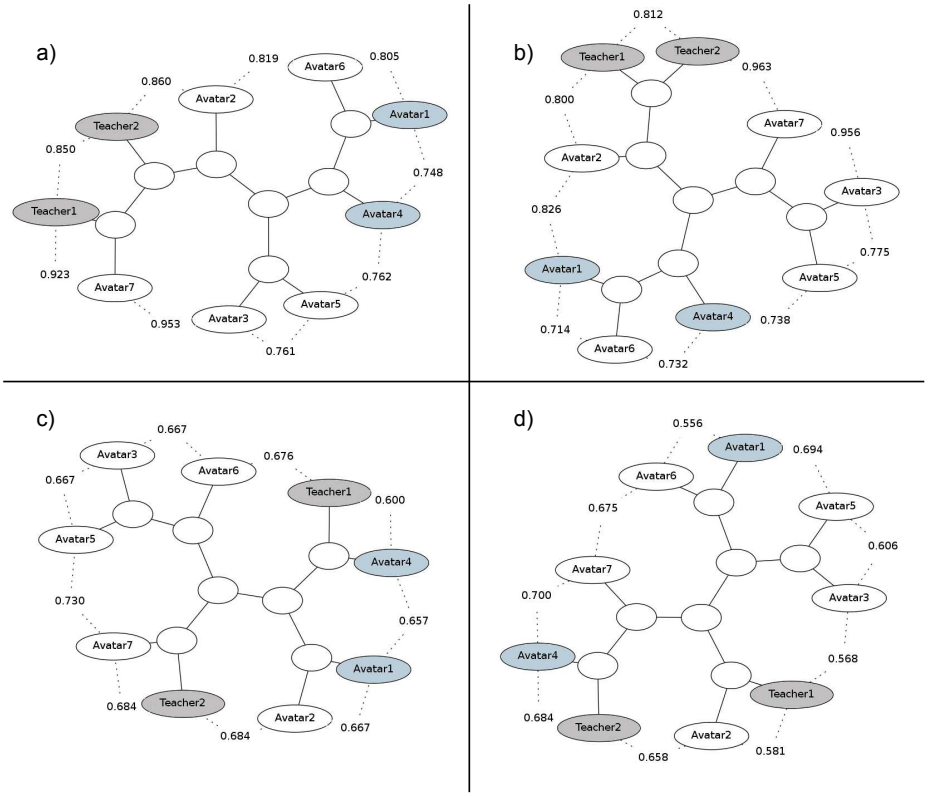


Figure 5.11: This figure shows the results of applying *NCD* over the behavioural files using the described representations. Representation #1 is show in subfigure *a*, representation #2 is shown in subfigure *b*. And subfigures *c* and *d* show representations #3 and #4 respectively. Two identified communities (*Teacher1* and *Teacher2*, and also *Avatar1* and *Avatar4*) has been highlighted.

dendrogram. This fact is shown in Subfigure *d* where *Avatar1* and *Teacher1* are located far away from their corresponding partners *Avatar4* and *Teacher2* respectively.

Also the loss of accuracy is reflected in the NCD-values shown between nodes in the dendograms. Representation #1 (Subfigure *a*) is the most accurate representation because it contains all the information in chronological order, i.e. it is the most complete representation. This fact is reflected with high NCD-values that goes from 0.75 to 0.95. The less accurate is the representation, the lower NCD-values are obtained. In this case, representation #4 is the representation with less information and this is shown with NCD-values between 0.55 and 0.7. It seems that files using representation #4 are similar, and this is a reasonable because all files contains the structure.

5.4.4.3 Using the proposed model for clustering avatars using their position

Finally, the last experiment tries to apply the proposed model for ACO algorithm to cluster the avatar position. In this case, a new problem appears. It is possible that K-means algorithm does not have enough data to perform the clustering task in a specific moment. For this reason, a time interval containing all data between two moments is used to guarantee enough data to k-means algorithm. Experimentally, this time interval (called *Snapshot*) was fixed to 40 seconds.

In order to test the proposed model, we have executed the K-means algorithm in Matlab using the selection of k proposed in [Pham et al., 2005]. Once we know the optimum k we have executed our model using this parameter and we have compared the results obtained by our model against the results obtained by Matlab.

The results obtained by our algorithm [Gonzalez-Pardo and Camacho, 2014a] [Gonzalez-Pardo et al., 2014c] are the same as the ones obtained by Matlab. There two reasons that can explain this behaviour:

- The number of students involved in the experiment is not really big. The lack of data can facilitate the clustering tasks.
- The number of clusters are fixed. This characteristic is not critical, but we could get more interesting results if we left to the ants the definition of the number of clusters.

Finally, Fig. 5.12 shows two graphical representations of the results from two different snapshots. These figures demonstrates that the proposed model can, also, be applied to clustering problems.

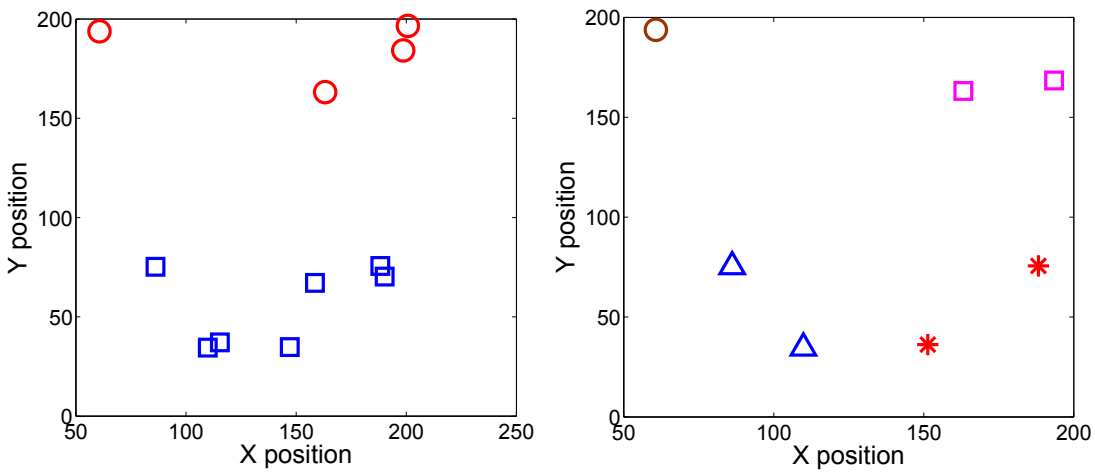


Figure 5.12: This figure shows the output of the clustering problem solved by the proposed model.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This chapter summarizes the main conclusions of this thesis and some future lines of work that defines the next research steps.

6.1 Conclusions

This thesis has been focused on the analysis of classical graph-based representations and its application in ACO algorithms into complex problems, and the development of a new ACO model that tries to take a step forward in this kind of algorithms.

The classical graph-based representation has been widely used in different works. Nevertheless, we have identify some problems that makes difficult the application of this representation in problems where real values are involved, or large problems. The specific problems identified are the following:

- The number of nodes that compose the problem is directly proportional to the number of variables and values that compose the problem. For example, in the case of *CSP* problems, the graph contains a node for each pair ⟨*variable,value*⟩ contained in the problem. If we are dealing with a clustering problem, the resulting graph contains as many nodes as pairs ⟨*instance,cluster*⟩ exists. Where *instance* is the set of input data, and *cluster* is one of the different clusters in which we want to group all the instances.
- The classical decision graphs are highly connected. This characteristic and the number of nodes created, could generate decision graphs whose size can be unmanageable.

The main advantage of classical graph-based representations consist in the simple behaviour of the ants that are executed into the graph. The ants only used to select the next node to visit, because the different assignations are encoded in each node of the graph.

In this thesis, a new model has been designed. Using the proposed model, the problem is represented using a reduced graph that affects to both, the ants behaviour, which becomes more complex, and the number of pheromones created in the system.

The new resulting graph contains a node for each variable in the problem. But the assignation of the value for each variable is performed by the ants. With the proposed model, the ants must select the next node to visit, and they need to select the corresponding value for this variable taking into account its local solution built so far. Therefore, the behaviour of the ants becomes more complex (and thus the structure that represent each ant requires more memory), but the behaviour can be still considered as "swarm" behaviour where simulations involving a large number of ants can be performed.

The main problem with the proposed model is related to the number of pheromones created in the graph. This number can grow so fast that the system could collapse. This is produced because the high number of nodes created in the classical graph-based representation is mapped to the number of pheromones needed during the execution of the algorithm. For this reason, a new metaheuristic, called *Oblivion Rate*, has been included in the model.

The goal of this metaheuristic is to control the number of pheromones created in the graph in order to avoid the saturation of the system. This metaheuristic is needed because the reduction in the size of the graph generates a more complex data structures to represent the phermones, and, also, the number of pheromones created in the system can grow very fast. These two consequences may produce the saturation of the system in terms of number of data structures created in the memory of the computer. The requirements that this metaheuristic must satisfy are the following:

1. The oblivion function must reduce the number of pheromones contained in the graph in order to avoid the system overload.

2. The number of pheromones must decrease gradually to allow the convergence of the system to the solution.

3. There must be always pheromones in the graph, because if in any step the system removes all the pheromones, the "*history*" of the execution will be lost, and it would be similar to launch the execution from the beginning.

4. In order to converge to good solutions, the system must remember the better pheromones, those with higher pheromone values.

In this thesis, three different kinds of Oblivion Rates have been designed and experimentally analyzed. The first one is an Oblivion Rate that uses a negative exponential function with a fixed exponent. In the second one, called *Dynamic Oblivion Rate*, the exponent of the negative exponential function is dynamic and it takes into account the saturation of the system, measured as the number of pheromones created divided by the total number of pheromones that can be created. The last Oblivion Rate used, called *CRO-based Oblivion Rate*, takes inspiration of a new bio-inspired algorithm (*Coral Reef Optimization*), that simulates the behaviour of several corals in the reef.

The proposed model has been applied to several complex problems. Initially, the *N-Queens Problems* revealed that the proposed model provides good solutions to the corresponding problem. These experiments shown that the evaporation rate is not useful for controlling the number of pheromones created in the system, and that the Oblivion Rate is needed. The evaporation rate decreases the pheromone values but it does not remove any pheromone from the graph.

The next application domain was the *Resource-Constraint Project Scheduling Problem* (RCPSP), where a set of jobs must be scheduled in such a way that the makespan (or time needed to execute all the jobs) is minimized. The experimental results reveal that the proposed model can be applied to this kind of problems, the best makespan obtained is close to the best makespan published by the research community. The main conclusion from these experiments is that the Dynamic Oblivion Rate provides an important reduction in the number of pheromones, maintaining the minimum makespan close to the optimum. In the problems analyzed in these experiments, the minimum reduction percentage in the number of pheromone was 79.75% (obtained with the problem *j60.sm*), whereas the maximum reduction was 97.85% for *m5.mm*.

As it can be observed, the quality of the found solutions (the mean quality and its standard deviation) is affected, only, by the complexity of the problem, and not by the Oblivion Rate functions used. Therefore, the reduction in the number of pheromones in the system does not affect to the quality of the solutions found by the algorithm.

Regarding the efficiency of the model, measured as the number of problems where our model reaches the best solution published by the research community, it seems that the dynamic, and the CRO-based, Oblivion Rates work quite similar. Nevertheless, we consider dynamic Oblivion Rate better than CRO-based Oblivion because the Dynamic Oblivion Rate does not need any parameter to be fixed, whereas the CRO-based needs from the definition of several parameters such as the size of the reef, the maximum number of attempts that new corals (pheromones) need to find a place in the reef, the predation probability, the predation percentage, etc.

The Oblivion Rate function is used to control, or reduce, the number of pheromones created in the system. This means that this metaheuristic is in charge of reducing the number of data structures created in memory. This reduction is needed because if it is not taken into account, the system can be saturated and the performance of the algorithm, measured as the time needed to solve the problem, get worst. Initially, the time needed to solve each problem was not a key measure of our system, because ACO is an heuristic algorithm. This means that it will need more time to get any solution than the standard algorithms, and also, that it is not guarantee that the algorithm finds the best solution of the problem. For this reason, we have not perform any comparison against any of the solvers available in the literature. Nevertheless, the use of the Oblivion Rate reduces significantly the time needed by the algorithm to complete the execution compared with the time needed by the ACO algorithm without using the Oblivion Rate. The most significant result is obtained when the system tries to solve the 100-*Queens*, and 200-Queens problems. In these cases, the system only completed the execution if the Oblivion Rate is used. The execution of a standard ACO algorithm, without using this metaheuristic, is not able to finish due to memory problems.

The third application domain was the well-known video game called *The Lemmings*. The goal of this new set of experiments was to understand what kind of algorithms is able to find more different paths from the starting point of the level to the exit point. In this case, a GA and the proposed model were compared. Regarding this set of experiments, we have compared the number of solutions because ACO is an heuristic algorithm (which means that it does not guarantee that the algorithm finds the best solution of the problem), and also, because the goal of these experiments is to test whether the propose model is able to find the path that connects the entry point with the exit point of the different levels. In the case where the proposed model is able to find the different paths, it can be used to automatically validate video game levels generated automatically. The results carried out in *The Lemmings* game determined that the algorithm

that finds more different solutions is our model. This result is produced because the construction of any individual in the GA is completely random without taking into account any contextual information related to the level.

Another experiment, carried out using *The Lemmings* game, studied how the representation of the ant colonies affects to the performance of the ACO algorithm. Two different kind of simulations (micro and macro) were performed. In the micro simulations each ant of the colony is used to represent a single lemming in the game, whereas each ant in the macro simulation represents the whole set of lemmings that compose the level. In these experiments, the macro simulations provides better results than the micro simulations in terms of the quality of the solutions found and number of different paths found. This is an expected result because the macro simulation approach performs more searches in parallel, at the same time, than the micro simulations.

The last application domain has been the avatar behaviour detection in Virtual Worlds. The goal of this experiment was to validate whether the proposed model can, also, be applied to cluster heterogeneous data. In this case, we have used the tracking data extracted from the interaction of several users in an educational virtual world (VirtUAM), to build the proposed reduced graph where the ACO algorithm is executed. These experiments reveals that the proposed approach can be, also, applied to clustering problems. Nevertheless, the low complexity of the dataset used in this work is not enough complex to apply the Oblivion Rate to control the number of pheromones created in the system.

As a general conclusion, we have identified a trade-off against the size of the resulting graph, and the complexity of the different data structures used in the model. The creation of a complex decision graph (like the ones proposed by Solnon, or Khan et. al.) generates simple data structures for representing each ant, and the pheromones. On the other hand, the simplification of the graph increases the complexity of the data structures of the ants and the pheromones. This complexity increase of the data structures, and the growth in the number of these structures, could saturate the system. Thus, the use of the Oblivion Rate metaheuristic is needed.

Finally, the proposed model solves one of the problems identified in the classical approaches. Using these approaches, the application domains that need to assign real values for the variables cannot be modelled. Although in this thesis, we have not solved any problem that uses real variables, the inclusion of these values only affects to the behaviour of the ants, that in the proposed approach are in charge of selecting the corresponding value for the different variables.

Tables 6.1 and 6.2 shows the relationship between the different goals of this thesis (described in Section 1.3), a brief description about the corresponding goal, and the published contributions generated in this thesis regarding to the specific goal. These contributions have been described in Chapter 8, where the acronym IJ means *International Journal*; IC means, *International Conference*; and finally, SIJ is used to represent the currently *Submitted International Journal*.

Goal	Description	Contribution	Cites
S1	Review of the State of the art related to Ant Colonies and Evolutionary Computation.	A wide review on swarm has been performed and it has been included in the different publications generated in this thesis.	See Chapter 8
S2	To identify strengths and weaknesses of the graph-based representations.	The main weaknesses of the classical graph-based representation is related to the size of the resulting decision graph where the ACO is executed. The main strength is the simple behaviour of the ants that only need to select the next node to visit.	(IC-1) (IC-2) (IC-4)
S3	To design a new graph-based representation	A new graph-based representation is proposed in this thesis. This representation creates a reduced graph that contains a node for each variable in the problem.	(IC-2) (IC-4)
S4	To analyze and design new ants behaviours.	The reduction on the complexity of the graph makes the behaviour of the ants more complex because, in the proposed model, ants have to select the next node to visit and the value for the corresponding variable encoded in the node.	(IJ-1) (SIJ-3) (IC-2) (IC-3) (IC-4)
S5	To study and design new metaheuristic to control the pheromone growth.	Different metaheuristics have been designed for controlling the number of pheromones. The initial study was perform using a negative exponential with a fixed exponent, and a negative exponential function with a dynamic exponent that takes into account the saturation of the system.	(SIJ-2) (IC-1) (IC-4)

Table 6.1: This table shows the first five thesis subgoals described in Section 1.3 and what is the contribution of this thesis regarding to each of them. The main publications originated from these contributions are shown in the last column.

Goal	Description	Contribution	Cites
S6	To study and design new metaheuristic inspired in GA.	Regarding this subgoal, a new metaheuristic based on the *Coral Reef Optimization* algorithm was design. The CRO is a GA-based algorithm that simulates the behaviour of the corals in a reef.	(SIJ-2) (IC-1)
S7	To define a new ACO model based on the graph-based representation.	The new model is created using the graph-based representation, the new behaviour of the ants and the Oblivion Rate metaheuristic.	(SIJ-1) (IC-2)
S8	To validate the new proposed model.	The proposed model has been validated using the *N-Queens* Problems.	(IC-4)
S9	To study and analyze the behaviour of the new model, and the pheromone control, in complex problems.	The proposed model has been tested using other application domains such as RCPSP, *The Lemmings* video game, and clustering avatar behaviour in VW.	(IJ-1) (IJ-2) (IJ-3) (SIJ-3) (IC-7)
S10	To compare the new ACO model against GA.	This comparison is performed using *The Lemmings* video game. Experimental results revealed that our model improve the performance of the GA, measured as the number of different solutions found and their quality.	(IJ-1) (IC-3) (IC-5)

Table 6.2: This table shows the last five thesis subgoals described in Section 1.3 and what is the contribution of this thesis regarding to each of them. The main publications originated from these contributions are shown in the last column.

6.2 Future lines of work

This section provides a brief description about the future lines of work that define the next research works. From the current state of this thesis, several issues or ”open questions” could be addressed:

1. The first line of work is related to the proposed model. We need try to improve both, the representation of the model and the metaheuristics designed for controlling the number of pheromones created in the system. In this thesis two different metaheuristics have been analyzed. On the one hand, the Dynamic Oblivion Rate is a really powerful metaheuristic and it does not contain any parameter to be fixed before the execution. The main problem with this metaheuristic is the definition of the maximum number of pheromones that can be created in the system. This definition can be really complex, or unknown, and its wrong definition can be crucial for the behaviour of the algorithm. On the other hand, the CRO-based Oblivion Rate works quite similar to the Dynamic Oblivion Rate but the main problem is that this metaheuristic depends on several parameters that must be fixed correctly to ensure the good performance of the metaheuristic. Some of these parameters are the size of the reef, the number of trials that each coral has to find a place in the reef, the percentage of corals that are candidates to die in each step, or the probability of the predation process.

2. Related to the previous line of work, a deep study about the correct configuration of the parameters involve in the model must be performed. In the previous line of work, the different parameters for the CRO-based Oblvion Rate has been listed, but the ACO algorithm has its own parameters like the number of ants in the algorithm, the number of steps, the evaporation rate, and the parameters α, and β, that defines the influence of the heuristic function and the pheromones in the decision process. All these parameters must be studied in order to understand the ranges of values in which the algorithm provides the best results, or to understand the range of values for these parameters depending on the kind of problem we want to solve.

3. To validate the proposed model, a comparison against the existing algorithm, platforms, or tools, from the state of the art must be done. This comparison can be focused in existing *Constraint Satisfaction Problem* solvers like Gecode [Gec, 2006] or Choco [cho, 2007], amongst others. What we have to do is to model the selected problems into Gecode or Choco solver, and compare the solutions provided by our model, and those provided by these platforms. This comparison must be performed in terms of quality, because the existing solver platforms for CSP problems uses optimized classical problems whereas our model uses an heuristic algorithm.

4. To study in depth, the performance of the proposed model in data mining problems. A deeper study of the behaviour of the proposed model to clustering techniques is required. There are several application domains related to clustering problem such as image recognition, or the identification of behavioural patterns, like stress, in real problems such as the training processes of *Unmanned Aircraft System* (UAS) [Ramirez-Atencia et al., 2014] operators where the utilization of the proposed approach could provide relevant results.

5. Finally, the application of the described model to other application domains is also needed. Two main application domains will be considered. On the one hand, problems like Timetable scheduling, Vehicle Routing Problem (VRP), WiFi network Design Problem (WifiDP), or the data analysis for cyber-security. On the other hand, other videogames like tower defense games [Palero et al., 2014], First-Person Shooter (FPS) [McPartland and Gallagher, 2012], or the application of our model in open videogames platforms as those described in Section 2.2.2.

CHAPTER 7

CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se exponen las conclusiones principales de esta tesis y algunas de las futuras líneas de trabajo que definen los siguientes investigaciones.

7.1 Conclusiones

Esta tesis se ha centrado en el análisis de la representación basada en grafos de problemas complejos para la utilización de algoritmos de colonia de hormigas (ACO, del inglés *Ant Colony Optimization*), y el desarrollo de un nuevo modelo de ACO que intenta mejorar los algoritmos de ACO existentes.

La representación clásica basada en grafos ha sido ampliamente usada en diversos trabajos. Sin embargo, hemos identificado algunas debilidades que dificultan la aplicación de dicha representación en problemas que utilicen valores reales, o en problemas grandes. Las diferentes debilidades encontradas se listan a continuación:

- El número de nodos que componen el problema es directamente proporcional al número de variables y valores que exiten en el problema. Por ejemplo, en el caso de los problemas de satisfacción de restricciones (CSP, del inglés *Constraint-Satisfaction Problems*), el grafo contiene un nodo por cada par ⟨*variable,valor*⟩ existente en el problema. Si estamos resolviendo un problema de clustering, el grafo resultante contiene tantos nodos como pares ⟨*instancia,cluster*⟩. Donde *instancia* es cada uno de los elementos que del conjunto de entrada, y *cluster* es cada uno de los diferentes clusters (o grupos) en los que se quieren agrupar todas las instancias.
- Los grafos de decisión tienen una alta conectividad. Esta característica, junto con el número de nodos que se crean, pueden crear grafos de decisión cuyo tamaño sea inmanejable.

La principal ventaja de esta representación clásica radica en la simpleza del comportamiento de las hormigas que se ejecutan en el grafo. Estas hormigas solo tienen que seleccionar el siguiente nodo que van a visitar, ya que los nodos en sí contienen la asignación específica.

En esta tesis, se ha propuesto un nuevo modelo. Usando este nuevo modelo, el problema se representa usando un grafo de tamaño más reducido que en el enfoque clásico. Dicha reducción afecta al comportamiento de las hormigas que se vuelve más complejo, y al número de feromonas que se crean en el sistema.

El nuevo grafo resultante contiene un nodo por cada una de las variables del problema. Pero la asignación de lo valores para cada variable es realizada por las hormigas. Con el modelo propuesto, las hormigas deben seleccionar el siguiente nodo que van a visitar, y además, seleccionarán valores para las variables codificadas en los nodos teniendo en cuenta su solución local construida hasta ese momento. Por lo tanto, el comportamiento de las hormigas se vuelve más complejo (y por tanto la estructura que representa a cada hormiga necesita más espacio en memoria), pero aún así, el comportamiento se puede considerar de "*swarm*" ya que se pueden realizar simulaciones masivas de dichos agentes.

El principal problema con el modelo propuesto está relacionado con el número de feronomas que se crean en el grafo. El número de feromonas puede crecer tanto que puede provocar la saturación, o el colapso, del sistema. Esto se produce porque el alto número de nodos que se creaban en el grafo con el enfoque clásico se ha traducido en posibles feromonas en el modelo propuesto. Por esta razón, una nueva metaheurística, llamada *Oblivion Rate*, ha sido incluida en el sistema.

El objetivo de esta metaheurística es el de controla el número de feromonas que se han creado en el grafo para evitar la saturación del sistema. Esta metaheurística es necesaria porque la reducción en el tamaño del grafo supone la creación de estructuras más complejas para representar tanto las hormigas, como las feromonas, y además, el numero de feromonas crece de manera muy rápida. Estas dos consecuencias pueden producir la saturación del sistema en términos de número de estructuras de datos creadas en la memoria del ordenador. Los requisitos que esta metaheurística debe cumplir son los siguientes:

1. La función de *Oblivion* debe reducir el número de feromonas creadas en el grafo para evitar la sobrecarga del sistema.

2. El número de feromonas debe decrecer gradualmente para permitir la convergencia del sistema hacia la solución.

3. Siempre debe de haber feromonas en el grafo, porque si en algún momento el sistema elimina todas las feromonas, la "*historia*" de la ejecución se pierde, y sería como ejecutar otra vez el algoritmo desde el principio.

4. Para que el sistema converja hacia buenas soluciones, se deben recordar las mejores feromonas, aquellas con valores más altos.

En esta tesis se han diseñado tres tipos diferentes de *Oblivion Rates*. La primera es una función exponencial negativa que usa un exponente fijo. La segunda función, llamada *Dynamic Oblivion Rate*, es una exponencial negativa cuyo exponente tiene en cuenta la saturación del sistema. Esta saturación se mide como número de feromonas creadas divido entre el número máximo de feromonas que se pueden crear. La última metaheurística diseñada, llamada *CRO-based Oblivion Rate*, está inspirada en un nuevo algoritmo llamado Optimización de Arrecifes

de Corales (o CRO, del inglés *Coral Reef Optimization*) que simula el comportamiento de corales dentro de un arrecife.

El modelo propuesto ha sido aplicado a diversos problemas complejos. Inicialmente, la aplicación del modelo al problemas de las *N-reinas* reveló que el modelo consigue buenas soluciones. Estos experimentos demostraron que la tasa de evaporación no es útil para controlar el número de feromonas creadas en el sistema, y que la *Oblivion Rate* es necesaria. Esto se debe a que la tasa de evaporación decrementa el valor de las feromonas pero no elimina ninguna feromona del grafo.

El siguiente dominio de aplicación fue la Planificación de proyectos con recursos limitados (o RCPSP, del inglés *Resource-Constraint Project Scheduling Problem*), donde un conjunto de tareas deben ser planificadas de tal manera que el *makespan* (o el tiempo necesario para ejecutar todas estas tareas) sea mínimo. Los resultados experimentales demuestra en modelo propuesto también puede ser utilizado a este tipo de problemas ya que los mejores *makespan* obtenido por nuestro modelo están cercanos a los mejores *makespan* publicados por la comunidad científica. La principal conclusión de este junto de experimentos es que la *Oblivion Rate* dinámica proporciona una importante reducción en el número de feromonas mientras que el makespan mínimo se mantiene cercano al óptimo. En el conjunto de problemas analizados, el mínimo porcentaje de reducción en el número de feromonas fue del 79.75% (obtenido con el problema *j60.sm*), mientras que el máximo porcentaje fue del 97.85% para el problema *m5.mm*.

Como se puede observar, la complejidad del problema afecta a la calidad de las soluciones encontradas (la calidad media y la varianza), pero la *Oblivion Rate* no afecta la calidad de dichas soluciones. Por lo tanto, la reducción en el número de feromonas en el sistema no afecta a la calidad de las soluciones que encuentra el algoritmo.

En cuanto a la eficiencia del modelo, medida como el número de problemas en los que nuestro modelo alcanza la mejor solución publicada por la comunidad científica, parace que la *Oblivion Rate* dinámica y la basada en CRO tienen un comportamiento parecido. Sin embargo, creemos que la *Oblivion Rate* dinámica es mejor que la basada en CRO debido a que la primera no tiene muchos parámetros que fijar, mientras que la *Oblivion Rate* basada en CRO necesita de la definición de diversos parámetros como el tamaño del arrecife, el número de intentos que tiene un coral (feromonas) para establecerse en el arrecife, la probabilidad de depredación, el porcentaje de depredación, etc.

La función de *Oblivion Rate* se usa para controlar, o reducir, el numero de feromonas creadas en el sistema. Esto significa que esta metaheurística se encarga de reducir el número de estructuras de datos creadas en memoria. Esta reducción es necesaria porque si no se tiene en cuenta, el sistema se puede saturar y el rendimiento del algoritmo, medido como el tiempo necesario para solucionar el problema, empeora. Inicialmente, el tiempo necesario para solucionar el problema no era un concepto clave ya que esta tesis se centra en el estudio del algoritmo heurístico ACO. Al ser un algoritmo heurístico, necesitará más tiempo que los algoritmos de búsqueda clásicos para encontrar una solución al problema, y ademas, no es seguro que la mejor solución se encuentre. Por esta razón, no hemos realizado ninguna comparación contra algunas de las herramientas existentes en la literatura para la resolución de estos problemas. Sin embargo, el uso de la *Oblivion Rate* reduce significativamente el tiempo necesario por el algoritmo para terminar su ejecución comparado con el tiempo necesario por un algoritmo de ACO estándar que no use *Oblivion Rate*. El resultado más significativo se ha obtenido cuando el sistema intenta solucionar el problema de las 100, o 200 reinas. En estos casos, el sistema termina la ejecución sólo

cuando la *Oblivion Rate* es utilizada. La ejecución del algoritmo estándar de ACO, sin *Oblivion Rate*, no es capaz terminar debido a problemas de memoria.

El tercer dominio de aplicación ha sido el videojuego llamado *Los Lemmings*. El objetivo de este nuevo conjunto de experimentos era entender qué tipo de algoritmo es capaz de encontrar mayor número de caminos diferentes desde el punto de entrada hasta el punto de salida del nivel. En este caso, se compararon un algoritmo genético (GA) y el modelo propuesto. En lo referente a este conjunto de experimentos, hemos utilizado el número de soluciones como métrica de comparación ya que ACO es un algoritmo heurístico (lo que significa que no está garantizado que el algoritmo encuentre la mejor solución al problema), y además, porque el objetivo de este junto de experimentos es demostrar si el modelo propuesto es capaz de encontrar el camino que conecta la entrada del nivel con la salida. Si nuestro modelo fuera capaz de encontrar dichos caminos, podría ser utilizado para la validación de pantallas generadas de manera automática. Los resultados obtenidos revelaron que nuestro modelo encontraba más soluciones diferentes que GA. Esto se debe a que la construcción de un nuevo individuo en GA is completamente aleatoria sin tener en cuenta la información contextual relacionada con el nivel.

En otro experimento realizado con el videojuego de *Los Lemmings* estudiamos cómo la representación de las hormigas afectan al rendimiento del algoritmo de hormigas. En este caso realizamos dos tipos diferentes de simulaciones, micro y macro. En las micro-simulaciones cada hormiga de la colonia representa un único lemming del juego, mientras que cada hormiga de la macro-simulación representa a todo el conjunto de lemmings que participan en el nivel. En estos experimentos, la macro-simulación proporciona mejores resultados que las micro-simulaciones en términos de calidad media de las soluciones encontradas, y número diferentes de caminos encontrados. Esto es un resultado esperado ya que el algoritmo de macro-simulación realiza más búsquedas en paralelo, al mismo tiempo, que las micro-simulaciones.

El último dominio de aplicación ha sido el *clustering*, o agrupamiento, del comportamiento de avatares en VirtUAM. El objetivo de estos experimentos fue validar si el modelo propuesto puede ser aplicado a tareas de *clustering* de datos heterogéneos. En este caso, hemos utilizado datos extraídos de las interacciones de diversos usuarios en mundos virtuales educativos para construir el modelo de grafo reducido donde el algoritmo de ACO es ejecutado. Estos experimentos demostraron que el modelo propuesto también puede ser utilizado para tareas de *clustering*. Sin embargo, el conjunto de datos usados en este trabajo no es suficientemente extenso para la utilización de la función de *Oblivion Rate* para controlar el número de feromonas creadas en el sistema.

Como conclusión general, se ha identificado un compromiso entre el tamaño del grafo de decisión y la complejidad de las estructuras de datos usadas en el modelo. La creación de un grafo de decisión complejo (como el propuesto por Solnon, o Khan et. al.) genera estructuras de datos simples para representar a las hormigas y a las feromonas. Por otro lado, la simplificación en el tamaño del grafo de decisión incrementa la complejidad de dichas estructuras de datos. Este incremento de la complejidad, y el crecimiento en el número de dichas estructuras, pueden saturar el sistema y por tanto, la utilización de la *Oblivion Rate* es necesaria.

Finalmente, el modelo propuesto soluciona uno de los problemas identificados en los enfoques clásicos. En estos enfoques, aquellos dominios de aplicación que necesiten de la utilización de variables reales, no pueden ser representados. Aunque en esta tesis no se han utilizado ningún dominio de aplicación que necesite de valores reales, la inclusión de dichos valores sólo afectaría al comportamiento al comportamiento de las hormigas que son las encargas de seleccionar el valor

Objetivo	Descripcion	Contribución	Cita
S1	Revisión del Estado del Arte relacionado con los algoritmos de Colonias de Hormigas y Computación Evolutiva.	La revisión sobre el estado del arte en estos algoritmos ha sido realizada, e incluída en las diferentes publicaciones generadas en esta tesis.	Ver Capítulo 8
S2	Identificar las fortalezas y las debilidades de las representaciones basadas en grafos.	La principal debilidad de la representación clásica basada en grafos está relacionada con el tamaño de grafo de decisión resultante donde el algoritmo de ACO se ejecuta. La principal fortaleza de esta representación radica en el comportamiento simple de las hormigas, que sólo tienen que decidir cuál será el siguiente nodo que visitarán.	(IC-1) (IC-2) (IC-4)
S3	Diseñar una nueva representación basada en grafo.	En esta tesis se ha propuesto una nueva representación basada en grafo que genera un grafo reducido. Este grafo contiene un nodo por cada variable del problema.	(IC-2) (IC-4)
S4	Analizar y diseñar nuevos comportamientos para las hormigas.	La reducción en la complejidad del grafo hace que el comportamiento de las hormigas sea más complejo, ya que en el modelo propuesto, las hormigas tienen que elegir el siguiente nodo que van a visitar y, además, deben seleccionar el valor para la variable representada en dicho nodo.	(SIJ-3) (IJ-1) (IC-2) (IC-3) (IC-4)

Table 7.1: En esta tabla se muestran los primero cuatro objetivos de esta tesis (descritos en la Sección 1.3) y la correspondiente contribución realizada en esta tesis. La última columna hace referencia a las publicaciones generadas referentes a dicho objetivo.

correspondiente para las variables representadas en cada nodo del grafo.

Para terminar, las Tablas 7.1 y 7.2 muestran la relación entre los diferentes objetivos de esta tesis (descritos en la Sección 1.3), una breve descripción sobre el correspondiente objetivo, y las contribuciones generadas en esta tesis referentes a dicho objetivo. Dichas contribuciones hacen referencia al índice de los trabajos en Capítulo 8, donde *IJ* hace referencia a *International Journal*, *IC* significa *International Conference*, y *SIJ* corresponde a las siglas de *Submitted International Journal*.

Objetivo	Descripcion	Contribución	Cita
S5	Estudiar y diseñar nuevas metaheurísticas para controlar el crecimiento de las feromonas.	Se han diseñado varias metaheuristicas para controlar el número de feromonas creadas. Para el estudio inicial, se ha utilizado una función exponencial negativa con un exponente fijo, y una función exponencial negativa con un exponente dinámico que tiene en cuenta la saturación del sistema.	(SIJ-2) (IC-1) (IC-4)
S6	Estudiar y diseñar nuevas metaheuristicas inspiradas en GA.	Respecto a este objetivo, se ha diseñado una nueva metaheurística basada en los algoritmos de *Optimización de Arrecifes de Corales* (CRO). Estos algorithmos simulan el comportamiento de los corales en el arrecife.	(SIJ-2) (IC-1)
S7	Definir un nuevo modelo de aco basado en la nueva representación basada en grafo.	Este nuevo modelo se ha diseñado usando la nueva representación basada en grafo, el nuevo comportamiento de las hormigas, y la metaheurística *Oblivion Rate*	(SIJ-1) (IC-2)
S8	Validar el nuevo modelo propuesto.	Esta validación se ha realizado utilizando el problema de las N reinas.	(IC-4)
S9	Estudiar y analizar el comportamiento del modelo, y el control de feromonas, en problemas complejos.	El modelo propuesto ha sido aplicado en diferentes dominios de aplicación como el RCPSP, el videojuego de *Los Lemmings*, y problemas de clustering de avatares en mundos virtuales.	(IJ-1) (IJ-2) (IJ-3) (SIJ-3) (IC-7)
S10	Comparar el nuevo modelo de ACO y los algoritmos genéticos.	Esta comparación se ha realizado usando el videojuego de *Los Lemmings*. Los resultados experimentales revelan que nuestro modelo mejora el rendimiento del GA, medido como el número de soluciones diferentes y calidad de dichas soluciones.	(IJ-1) (IC-3) (IC-5)

Table 7.2: En esta tabla se muestran los últimos cinco objetivos de esta tesis (descritos en la Sección 1.3) y la correspondiente contribución realizada en esta tesis. La última columna hace referencia a las publicaciones generadas referentes a dicho objetivo.

7.2 Líneas de Trabajo Futuro

Esta sección proporciona una descripción sobre las futuras líneas de trabajo que definen los siguientes pasos de investigación. Partiendo del estado actual de esta tesis, hay varias cuestiones que deberán ser analizadas:

1. La primera línea de trabajo está relacionada con el modelo propuesto. Tenemos que mejorar tanto la representación del modelo como la metaheurística diseñada para controlar el número de feromonas que se crean en el sistema. En esta tesis dos metaheurísticas han sido analizadas. La primera de ella, llamada *Oblivion Rate* dinámica, es una metaheurística muy potente que no necesita de la definición de ningún parámetro antes de su ejecución. El principal problema con esta metaheurística es la definición del número máximo de feromonas que pueden ser creadas en el sistema. Esta definición puede ser compleja, o desconocida, y su incorrecta definición puede ser crucial para el comportamiento del algoritmo. La segunda metaheurística, llamada *CRO-based Oblivion Rate*, funciona similar a la metaheurística anterior pero el principal problema de esta segunda *Oblivion Rate* es que depende de numerosos parámetros que deben ser fijados correctamente para asegurar el buen funcionamiento de la metaheurística. Algunos de estos parametros son el tamaño del arrecife, el número de intentos que tiene un coral (feromona) para establecerse en el coral, el porcentaje de corales que son candidatos al proceso de depredación, o la probabilidad de depredación.

2. Relacionado con el trabajo de futuro previamente descrito, se debe realizar un estudio detallado sobre la correcta configuración de los parámetros involucrados en el modelo. En la línea de trabajo anterior, se han enumerado los diferentes parámetros que forman parte de la *Oblivion Rate* basada en CRO, pero el algoritmo de ACO también tiene sus propios parámetros como el número de hormigas que participan en el algoritmo, el número de pasos que se ejecutará el algoritmo, la tasa de evaporación, y los parámetros α, y β, que definen la influencia de la función heurística y las feromonas en el proceso de decisión de las hormigas. Todos estos parámetros ben ser estudiados para entender los rangos de valores en los cuales el algoritmo proporciona los mejores resultados, o para entender los mejores rangos de valores para estos parámetros dependiendo del tipo de problema a solucionar.

3. Se debe validar el modelo propuesto comparando contra los algoritmos, plataformas, o herramientas existentes en el estado del arte. Esta comparación puede estar enfocada en programas para la resolución de problemas de CSP como Gecode [Gec, 2006], o Choco [cho, 2007], entre otros. Debemos modelar el problema seleccionado dentro de Gecode o Choco, y comparar las soluciones generadas por nuestro modelo, y aquellas generadas por dichas plataformas. Esta comparación deberá realizarse en terminos de calidad de las soluciones, ya que las plataformas para resolver problemas de CSP usan métodos clásicos optimizados mientras que nuestro modelo utiliza un algoritmo heurístico.

4. Estudio en profundidad, del funcionamiento del modelo propuesto en problemas de minería de datos. Existen diversos dominios de aplicación relacionados con problemas de *clustering* tales como el reconocimiento de imágenes, o la identificación de patrones de comportamiento (como el estres) en problemas reales como los procesos de entrenamiento de operadores de Sistemas Voladores no Tripulados (o UAS, del inglés *Unmanned Aircraft System*) [Ramirez-Atencia et al., 2014] donde la utilización del modelo propuesto puede proporcionar resultados significativos.

5. Por último, se estudiará la aplicación del modelo propuesto en otros dominios de aplicación. De momento, se considerarán dos dominios de aplicación. Por un lado problemas como la planificación de horarios, el problema de enrutamiento de vehículos (o VRP, del inglés *Vehicule Routing Problem*), el diseño de redes wifi (o WifiDP, del inglés *WiFi network Design Problem*) o el análisis de datos para ciberseguridad. Por otro lado, otros videojuegos como los juegos de defensa [Palero et al., 2014], o *First-Person Shooter* (FPS) [McPartland and Gallagher, 2012], o la aplicación de nuestro modelo en plataformas de videojuegos abiertas como aquellas descritas en la Sección 2.2.2.

CHAPTER 8

CONTRIBUTIONS

During the development of this work several publications have been generated. This chapter provides a short description of them, and the sections and chapters where these contributions can be found in the thesis. Finally, these publications have been organized by journals and conferences, and sorted by year.

International Journals

(IJ-1) A. Gonzalez-Pardo, F. Palero, D. Camacho: *An Empirical Study on Collective Intelligence Algorithms for Video Games Problem-Solving*. Computing and Informatics. In press. (2014). Impact factor = 0.254 (JCR, 2012).

- Short summary: Constrained Satisfaction Problems (CSP) have taken an important attention from the research community due to their applicability to real problems. Any CSP problem is usually modelled as a constrained graph where the edges represents a set of restrictions that must be verified by the variables (represented as nodes in the graph) which will define the solution of the problem. This paper studies the performance of two particular CI algorithms, Ant Colony Optimization (ACO) and Genetic Algorithms (GA), when dealing with graph-constrained models in video games problems. As application domain, it has been selected the "Lemmings" video game, where a set of *lemmings* must reach the exit point of each level. In order to do that, each level is represented as a graph where the edges store the allowed movements inside the world. The goal of the algorithms is to assign the best skills in each position on a particular level, to guide the *lemmings* to reach the exit. The paper describes how the ACO and GA algorithms have been modelled and applied to the selected video game. Finally, a complete experimental comparison between both algorithms, based on the number of solutions found and the levels solved, is analysed to study the behaviour of those algorithms in the proposed domain.
- Contribution: This work was the first deep work about *The Lemmings* video game and allow us to understand the difference between the proposed model and a GA in the task of finding the different path to solve a level. This contribution is related to Sections 4.1.3, 4.2.2 and 5.3.1.

(IJ-2) A. Gonzalez-Pardo, A. Rosa, D. Camacho: *Behaviour-based identification of student communities in Virtual Worlds*. Computer Science and Information Systems, Vol. 11, No. 1, pp. 195–213. (2014).
Impact factor = 0.549 (JCR, 2012).

- Short summary: Using a VW platform called VirtUAM, the information extracted from different experiments are used to analyse and define students communities based on their behaviour. To define the individual student behaviour, different characteristics are extracted from the system, such as the avatar position (in form of GPS coordinates) and the set of actions (interactions) performed by students within the VW. Later this information is used to automatically detect behavioural patterns. This paper shows how this information can be used to group students in different communities based on their behaviour. Experimental results show how community identification can be successfully perform using K-Means algorithm and *Normalized Compression Distance*. Resulting communities contains users working in near places or with similar behaviours inside the virtual world.
- Contribution: This work has contributed to this thesis by providing the basic concepts about communities evolution. This evolution is needed to track the user behaviours in virtual worlds. This contribution is related to Section 5.4.

(IJ-3) A. Berns, A. Gonzalez-Pardo, D. Camacho: *Game-like language learning in 3-D Virtual environments. Computers & Education*, Vol. 60, No. 1, pp. 210-220. (2013).
Impact factor = 2.775 (JCR, 2012).

- Short summary: This paper presents our recent experiences with the design of game-like applications in 3-D virtual environments as well as its impact on student motivation and learning. Therefore our paper starts with a brief analysis of the motivational aspects of videogames and virtual worlds (VWs). We then go on to explore the possible benefits of both in the area of foreign language learning. For our research study we have designed a VW-platform, called VirtUAM. This permits us to store and record data related to users' behaviour within the VW. Furthermore the platform has been employed to build several islands (virtual spaces), which implement different game levels. The virtual spaces themselves are used to give students a basic training in different language skills related to the German language. In order to obtain data regarding the game's impact on student learning and motivation, we designed several tests, which were completed both before and after the student participants played the game. Additionally we gave them a general questionnaire, which was only filled out after the game and which aimed at getting personal feedback from the learners. Quantitative and qualitative results shown in this work are part of a larger project which intends to study the impact of game-like applications within virtual environments and with regard to teaching and learning processes in general.
- Contribution: The content of this paper is directly related to Section 5.4.

(IJ-4) A. Gonzalez-Pardo, P. Varona, D. Camacho, F. de Borja Rodríguez Ortiz: *Communication by identity discrimination in bio-inspired multi-agent systems*. Concurrency and Computation: Practice and Experience, Vol. 24, No. 6, pp. 589-603. (2012).
Impact factor = 0.845 (JCR, 2012).

- Short summary: Network communications have been widely studied in the last decades in different research fields: Artificial Intelligence, Computer Science, Biology, Medicine and Psychology amongst others. Some important efforts have been carried out to analyze communication features such as overhead, connectivity, or communication protocols in these areas from their own perspectives. When this problem is restricted to Intelligent Agents or Multi-Agent Systems (MAS), networks are built by a set of interconnected agents that can be software or hardware. In MAS, communication optimization is used to improve the overall performance of the system by reducing the information sharing (i.e. number of messages or message size) between the agents. This paper analyzes a scale free network topology of agents to solve a multi-sorting problem. The agents use their local information as well as a bio-inspired identity discrimination process, to select only those messages that are relevant for each agent to solve jigsaw puzzles. We provide a comprehensive study on the influence of some essential parameters (memory information size and reconnection probability) in an agent network, and how they can be set to obtain a better performance in the system. The experiments show that this strategy contributes to reduce the number of iterations needed to solve the problem.
- Contribution: This work presents a study about how the quantity information exchanged amongst several agents, influences to the quality of the solutions found by the Multi Agent System. This work represents an initial study that is related to the control of pheromone, because controlling the number of pheromones in the system can be considered equivalent (or at least related) to control the quantity of information exchange amongst the ants. This paper is related to Sections 5.3.1.2 and 5.3.2.

International Conferences

(IC-1) A. Gonzalez-Pardo, D. Camacho: *Solving Resource-Constraint Project Scheduling Problems based on ACO algorithms*. Ninth International Conference on Swarm Intelligence (ANTS 2014). Brussels, Belgium. In press. **Core-ERA B**.

- Short summary: *Constraint Satisfaction Problems* (CSP) provide an interesting complex framework to test and prove new heuristic-based solving methods approaches like Swarm Intelligence. The interest is related to both, the complexity of CSP problems (*NP-complete*) and its possible application to real and industrial problems. This paper applies an ACO algorithms with a new CSP-graph based representation to solve Resource-Constrained Project Scheduling Problems (RCPSP). To control the fast growing number of pheromones created in the graph, a new meta-heuristic, called *Oblivion Rate*, has been designed. A complete experimental analysis has been carried out using several RCPSP datasets to evaluate the quality of solutions found when this approach is applied.
- Contribution: This paper studies the influence of the Dynamic Oblivion Rate for solving *Resource-Constraint Project Scheduling Problems*. All the Single-Mode

problems extracted form PSPLib library have been analyzed, and the results are compared against the performance of the system without using any Oblivion Rate. This paper is related to Sections 3.2, 4.1.2, 4.2.3, 4.3.2.2 and 5.2.2.

(IC-2) A. Gonzalez-Pardo, D. Camacho: *A New CSP Graph-Based Representation to Resource Constrained Project Scheduling Problem*. IEEE Congress on Evolutionary Computation (IEEE CEC 2014). July, Beijing, China. In press. **Core-ERA A**.

- Short summary: This work is an extension of a previous one, where a new CSP graph-based representation to solve Constraint Satisfaction Problems (CSP) by using Ant Colony Optimization (ACO) were proposed. This paper studies the behaviour of the CSP graph-based representation when it is applied to a real-world complex problem, in this case the RCPSP. The dataset used in this work has been extracted from Project Scheduling Problem Library (PSPLIB). Experimental results show that the proposed approach provides excellent results, closer to the optimum values published in the PSPLIB repository. Also, it has been analysed how the number of jobs and the number of different execution modes affect the performance of the algorithm.
- Contribution: This work represents the first paper where the proposed model was applied to RCPSP problems. This initial study help us to identify the different parameters that influences in the complexity of a RCPSP problem. The contribution of this paper is described in Sections 3.2, 4.1.2, 4.2.3, 4.3.2.2 and 5.2.1.

(IC-3) A. Gonzalez-Pardo, F. Palero, D. Camacho: *Micro and Macro Lemmings simulations based on ants colonies*. Evostar. EvoGames 2014. April, Granada, Spain. In press. **Core-ERA B**.

- Short summary: The paper describes both the graph model and the context-based heuristic, designed to implement our ACO approach. Afterwards, two different kind of simulations have been carried out to analyse the behaviour of the ACO algorithm. On the one hand, a micro simulation, where each ant is used to model a lemming, and a macro simulation where a swarm of lemmings is represented using only one ant. Using both kind of simulations, a complete experimental comparison based on the number and quality of solutions found and the levels solved, is carried out to study the behaviour of the algorithm under different game configurations.
- Contribution: In this work the influence of the micro, and macro, simulation for ACO algorithms applied to *The Lemmings* video game is presented. This work is described in Sections 4.1.3, 4.2.2 and 5.3.2.

(IC-4) A. Gonzalez-Pardo, D. Camacho: *A new CSP graph-based representation for Ant Colony Optimization*. IEEE Congress on Evolutionary Computation (IEEE CEC 2013). June, Cancun, Mexico. pp.689-696. **Core-ERA A**.

- Short summary: This paper presents a new efficient CSP graph-based representation to solve CSP by using Ant Colony Optimization (ACO) algorithms. This paper presents also a new heuristic (called *Oblivion Rate*), that have been designed to improve the current state-of-the-art in the application of ACO algorithms on these domains. The presented graph construction provides a strong reduction in both, the number of connections and the number of nodes needed to model the CSP. Also, the new heuristic is used to reduce the number of pheromones in the system (allowing to solve problems with an increasing complexity). This new approach has been tested, as case study, using the classical N-Queens Problem. Experimental results show how the new approach works in both, reducing the complexity of the resulting CSP graph and solving problems with increasing complexity through the utilization of the *Oblivion Rate*.
- Contribution: The contribution of this work is directly related to Sections 3.3, 4.1.1, 4.2.1, 4.3.1, 4.3.2.1, and 5.1.

(IC-5) A. Gonzalez-Pardo, D. Camacho: *Environmental Influence in Bio-inspired Game Level Solver Algorithms*. Intelligent Distributed Computing VII (IDC 2013). September, Prague, Czech Republic. Vol. 511, pp. 157-162.

- Short summary: This paper studies how Genetic Algorithms (GA) and Ant Colony Optimization (ACO) algorithms can be applied to automatically solve levels in the well known Lemmings Game. The main goal of this work is to study the influence that the environment exerts over these algorithms, specially when the goal of the selected game is to save an individual (lemming) that should take into account their environment to improve their possibilities of survival. The experimental evaluations carried out reveals that the performance of the algorithm (i.e. number of paths found) is improved when the algorithm uses a small quantity of information about the environment.
- Contribution: The contribution of this work is described in Section 5.3.1.

(IC-6) A. Gonzalez-Pardo, D. Camacho: *Maximal Component detection in graphs using swarm-based and genetic algorithms*. Intelligent Distributed Computing VI (IDC 2012). September, Calabria, Italy. Vol. 446, pp. 247-252.

- Short summary: This work describes how these two different optimization strategies (GA and ACO) can be adapted and used to extract the different sub-graphs that contains the maximum number of nodes. Experimental results show the best results are obtained using ACO algorithm, but new strategies must be taken into account in order to improve the results.
- Contribution: This paper provides an initial comparison of both algorithms (GA and ACO) applied to solving problems, in this case, the maximal component detection in graphs. The contribution of this paper is related to Section 5.3.1

(IC-7) A. Gonzalez-Pardo, F. de Borja Rodríguez Ortiz, E. Pulido, D. Camacho: *Using virtual worlds for behaviour clustering-based analysis*. ACM Multimedia 2010. ACM Workshop on Surreal Media and Virtual Cloning. October, Florence, Italy. ACM Press, pp. 9 – 14. **Core-ERA A***

- Short summary: This paper studies how the avatars position, what they are looking at (eye-gazing) or what they are talking about, can be integrated in order to apply clustering techniques. Monitoring avatars in a virtual world is a useful task that allows the identification of behavioural groups. The meaning of these groups depends on the application domain, for example in educational virtual worlds, they can represent whether students are paying attention to the teacher's explanation or not.
- Contribution: The contribution of this work is related to Section 5.4.

Submitted International Journals

(SIJ-1) A. Gonzalez-Pardo, D. Camacho. *Solving Project Scheduling Problems through Swarm-based approaches*. International Journal of BioInspired Computation (IJBIC). Edited by Inderscience Publishers. Submitted May 2014.

(SIJ-2) A. Gonzalez-Pardo, D. Camacho. *A Study on Metaheuristic for Pheromone Control in ACO Algorithms*. International Journal of Neural Systems (IJNS). Edited by Wold Scientific. Submitted May 2014.

(SIJ-3) A. Gonzalez-Pardo, D. Camacho. *Designing an ACO Model for Clustering Student Behaviours in Virtual Worlds*. Integrated Computer-Aided Engineering (ICAE). Edited by IOS Press. Submitted June 2014.

APPENDIX A

RESULTS FOR PSPLIB DATASET

This appendix contains all the information regarding the experiments carried out in this thesis applied to PSPLib libray.

PSPLib is a set of *Resource-Constraint Project Scheduling Problems*. This library currently contains 12119 instances grouped in 24 different problems. In this thesis, 7564 instances belonging to 14 different problems have been analyzed. Table A.1 shows the characteristic for the different instances taken into account in this work.

Problem Name	Instances	Jobs	Modes	Renewable	Non Renewable
j30.sm	480	30	1	4	0
j60.sm	480	60			
j90.sm	480	90			
j120.sm	600	120			
j10.mm	536	10	3	2	2
j12.mm	547	12			
j14.mm	551	14			
j16.mm	550	16			
j18.mm	552	18			
j20.mm	554	20			
j30.mm	640	30			
m2.mm	481	16	2	2	2
m4.mm	555		4		
m5.mm	558		5		

Table A.1: This table contains the main characteristics of the different problems analyzed in this thesis. For each problem, this table shows the number of instances that compose the problem, the number of jobs that compose the project, the number of modes, and the number of renewable, and non renewable, resources for each job.

Problem	Normal ACO	Dynamic Oblivion	Reduction pct.
j30.sm	1352	225	83.35%
j60.sm	3789	767	79.75%
j90.sm	11262	1791	84.09%
j120.sm	24781	4813	80.57%
j10.mm	2023	68	96.63%
j12.mm	2695	83	96.92%
j14.mm	3246	101	96.88%
j16.mm	3841	120	96.87%
j18.mm	4315	142	96.71%
j20.mm	4865	168	96.54%
j30.mm	5922	276	95.33%
m2.mm	2092	112	94.64%
m4.mm	4891	268	94.52%
m5.mm	5645	121	97.85%

Table A.2: This table shows the maximum number of pheromones created in the system using a dynamic Oblivion Rate, and without the Oblivion Rate.

All these instances have been solved using a normal ACO, the proposed model using the Dynamic Oblivion Rate, and the proposed model using the CRO-based Oblivion Rate. Table A.2 shows the maximum number of pheromones created in the system with a normal ACO, and the number of pheromones created in the system by the proposed model using the Dynamic Oblivion Rate.

Finally, the evolution of the number of pheromones for each of the problems analyzed in this thesis are shown. The following figures shows the evolution of the number of pheromones when there is not any pheromone control procedure (Normal), when the model is using the Dynamic Oblivion Rate (Dynamic), and when the CRO-based Oblivion Rate is used to control the number of pheromones.

Note that the size of the coral reef for the CRO-based Oblivion Rate is fixed taking into account the maximum number of pheromones created in the system by the Dynamic Oblivion Rate. In this way, the comparison between both metaheuristic can be performed.

Figures A.1 and A.2 shows the evolution for the Single-Mode projects, i.e. *j30.sm*, *j60.sm*, *j90.sm* and *j120.sm*. Regarding to the Multi-Mode problems, Figures A.3, A.4 and A.5 shows the results for the *j10.mm*, *j12.mm*, *j14.mm*,*j16.mm*, *j18.mmm*, *j20.mm* and *j30.mm*. Finally, the Multi-Mode problems that differ in the number of execution modes, the *m2.mm*, *m4.mm*, and *m5.mm* problems, are shown in Figure A.6.

In all these figures, it can be clearly observed how the Oblivion Rate reduces drastically the number of pheromones in comparison to the normal execution of the ACO algorithm.

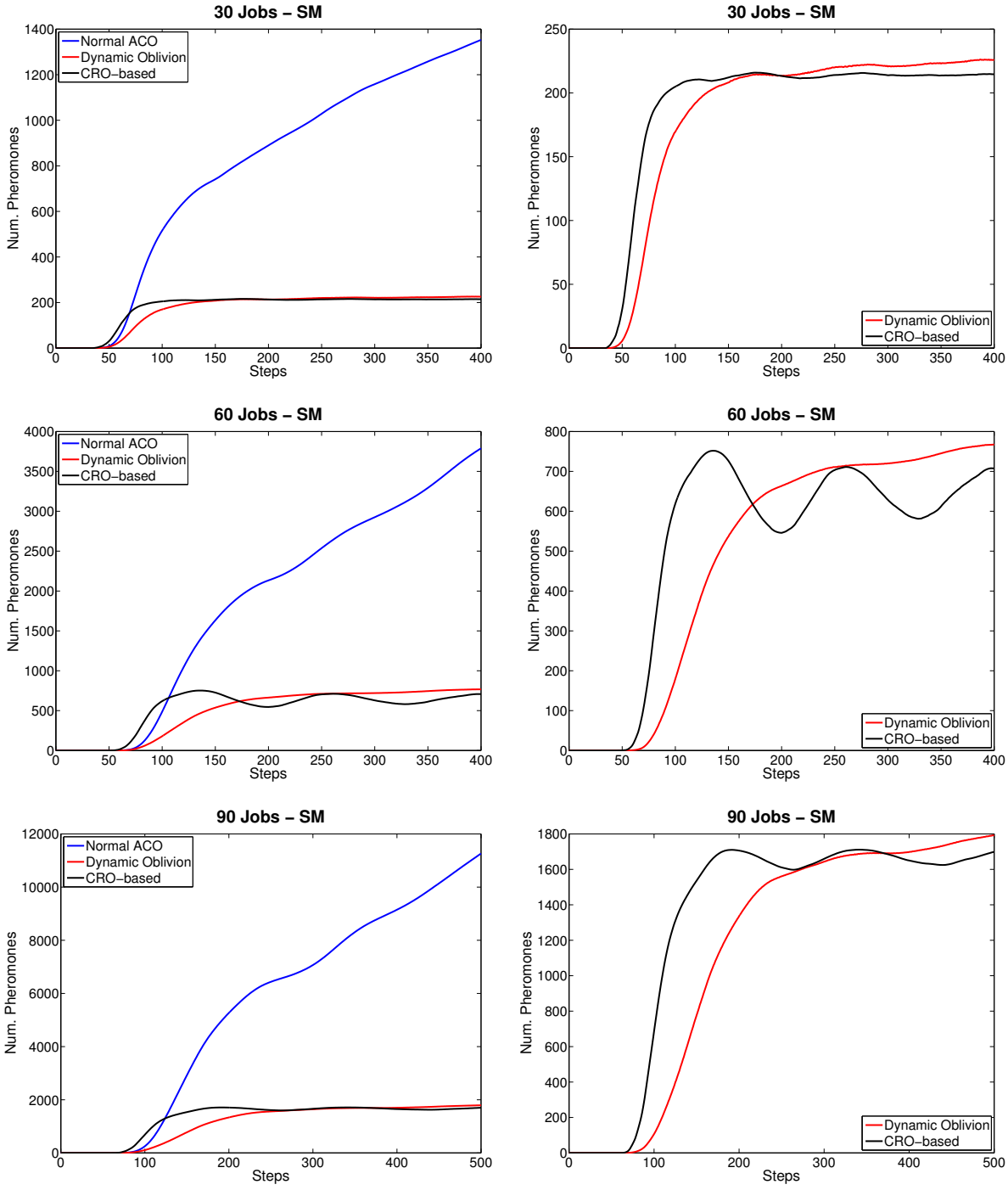


Figure A.1: Evolution of the number of pheromones created in the system for the problems *j30.sm*, *j60.sm* and *j90.sm*.

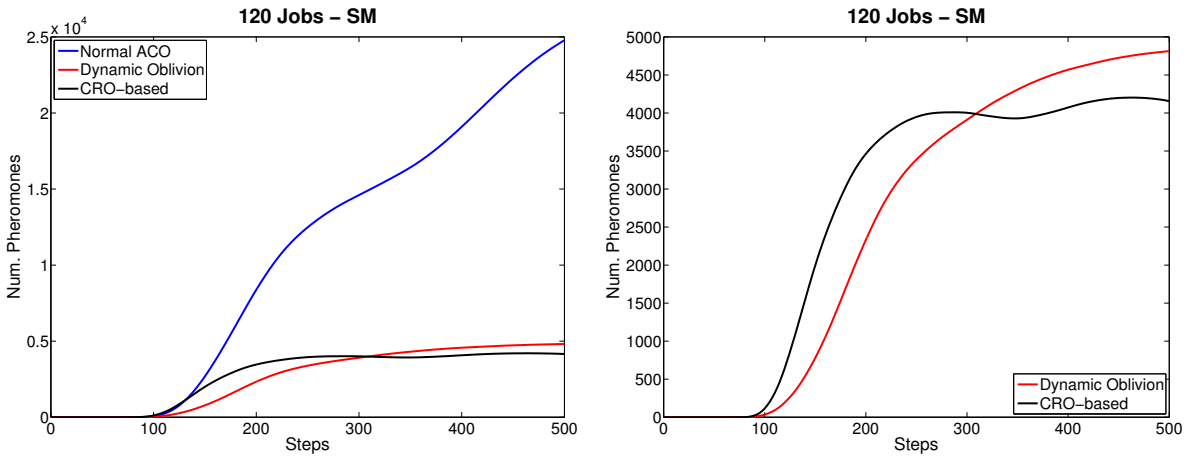


Figure A.2: This figure shows the evolution of the number of pheromones for the *j120.sm* problem.

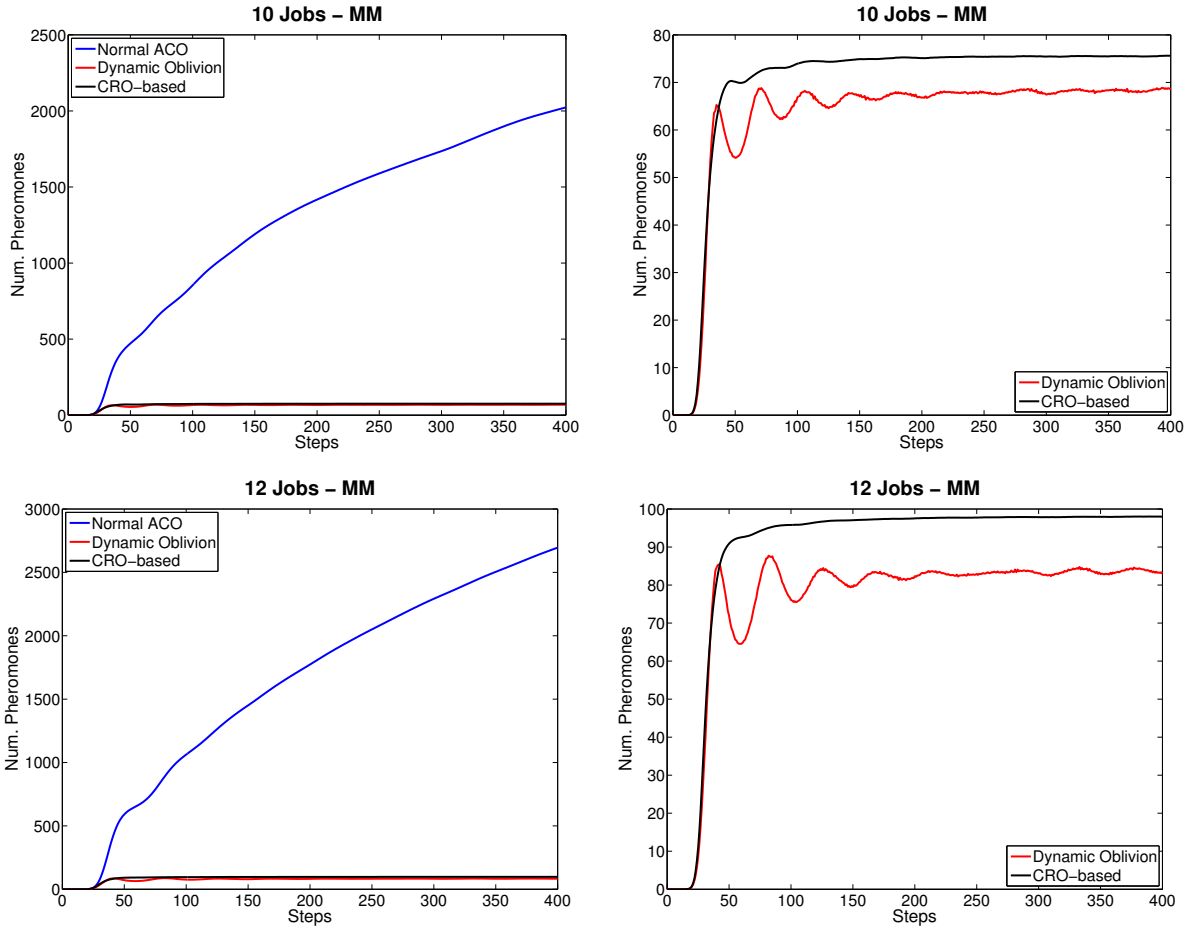


Figure A.3: Evolution of the number of pheromones for some of the Multi-Mode problems. This figure shows the results for *j10.mm* and *j12.mm*.

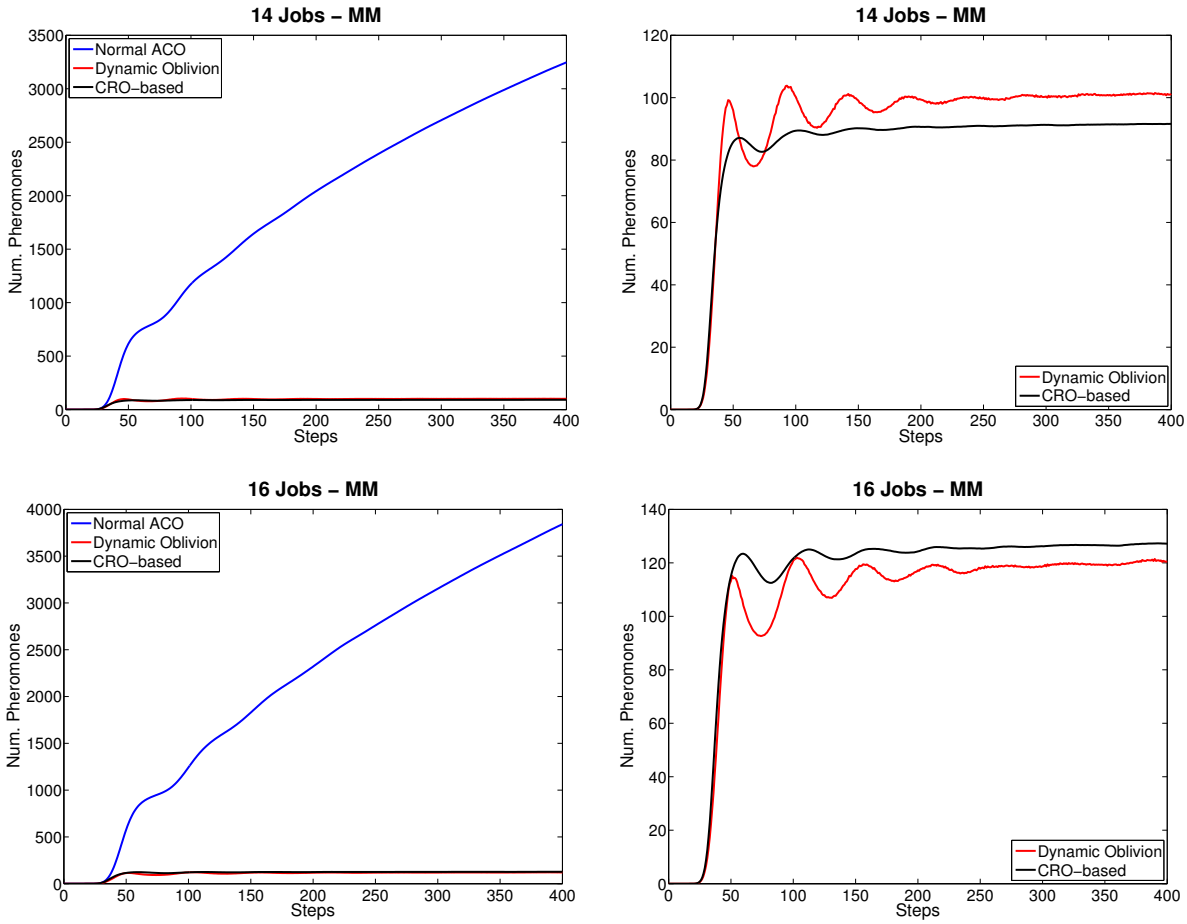


Figure A.4: This figure shows the evolution of the number of phermones created in the system for the problems *j14.mm* and *j16.mm*.

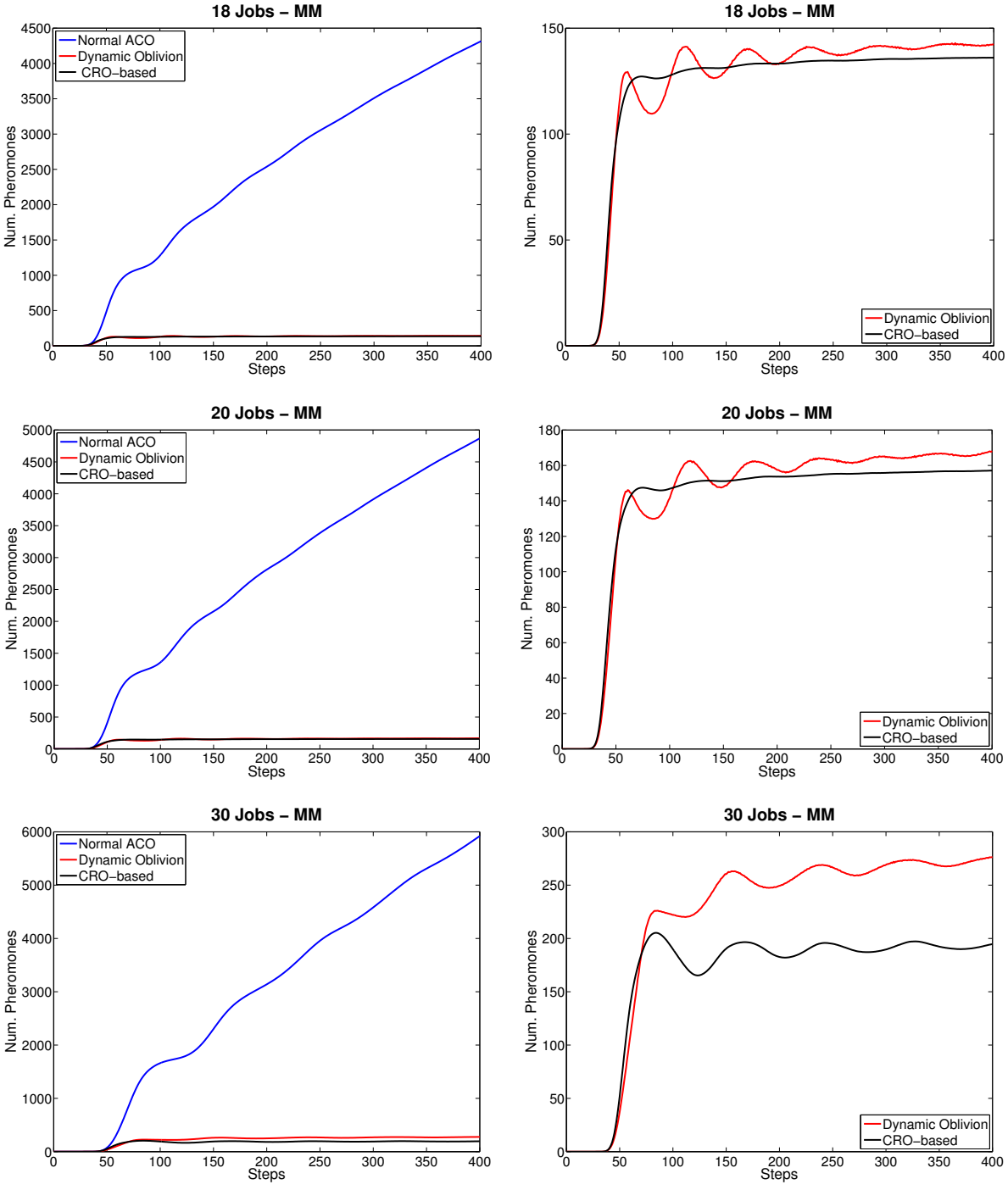


Figure A.5: This figures correspond to the number of pheromones created in the system for *j18.mm*, *j20.mm*, and *j30.mm* problems.

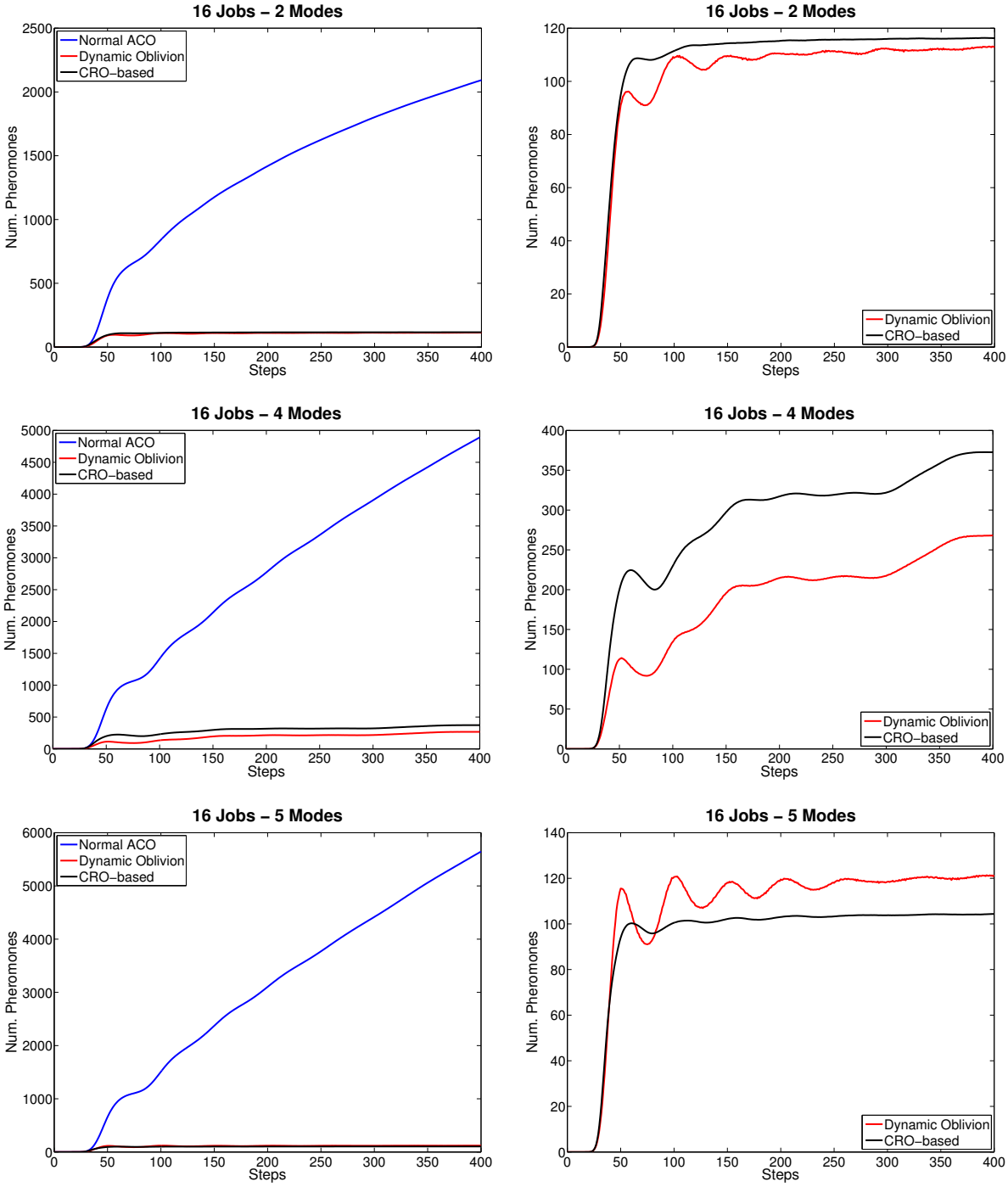


Figure A.6: In this figures, the evolution of the number of pheromones for *m2.mm*, *m4.mm* and *m5.mm* problem are shown.

APPENDIX B

THE LEMMINGS LEVELS

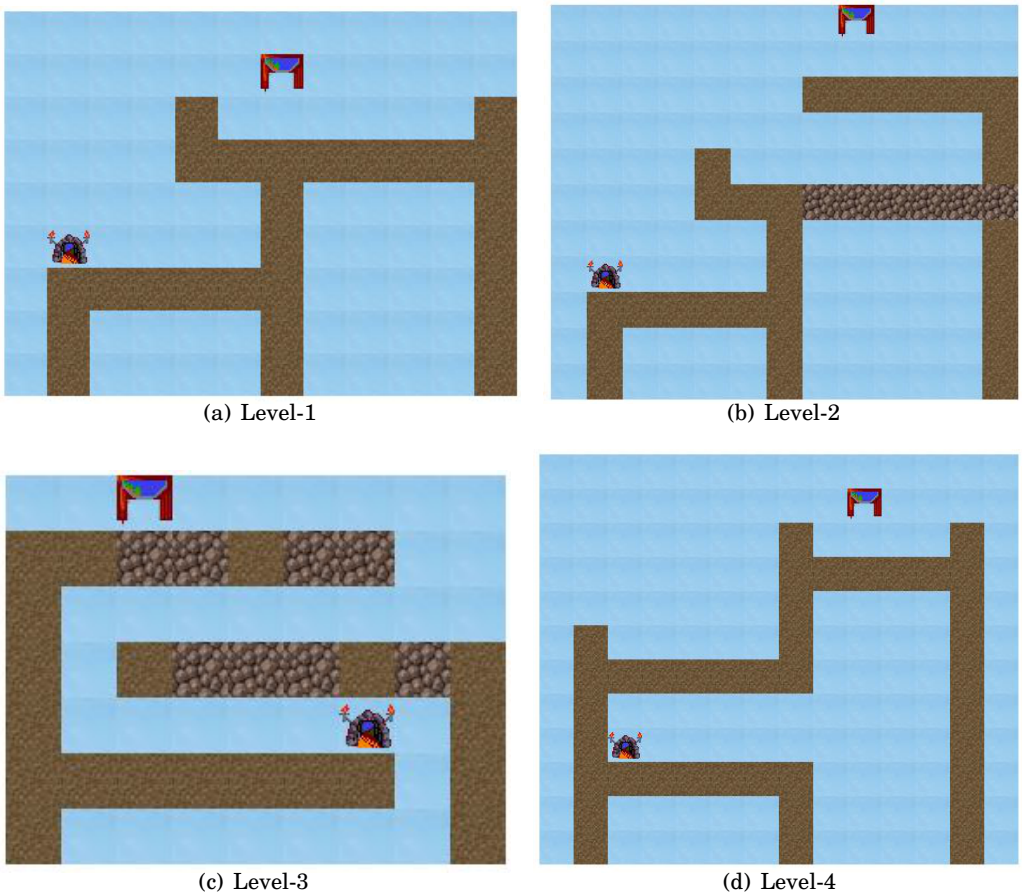
This appendix shows the 14 different levels for *The Lemmings* video game. These levels have been grouped, by hand, in easy, medium and hard levels taking into account the size of the levels, the terrain that composed each level, the number of lemmings to save, and the initial number of skills to assign to the lemmings:

- **Easy**. These levels use both kind of materials, and the human-likes solution is a short path (few lemmings actions) with few skills that are required to reach the exit. When non editable material is used, the lemmings colonies are "guided" to the exit because those skills related to "digging" abilities cannot be used (therefore the search space is reduced). Figure B.1(a) shows an example of an easy level.
- **Hard**. These type of levels only use editable materials, and the solution to reach the exit needs from a large number of skills and actions (large solution paths) to be taken. Figure B.3(b) shows the representation of one hard level.
- **Medium**. These levels use a combination of previous ones as Figure B.2(a) shows. In these kind of levels, both materials can be used and the solutions can be a mixture of actions. In the level, it is possible to find parts with a high level of freedom for the lemmings (they can use all of the available skills), and some other parts where the number of skills that can be used is reduced.

Table B.1 shows the complexity for each level designed in this thesis.

Level	Complexity	Figure
1	Easy	Fig. B.1(a)
2	Easy	Fig. B.1(b)
3	Easy	Fig. B.1(c)
4	Easy	Fig. B.1(d)
5	Medium	Fig. B.2(a)
6	Medium	Fig. B.2(b)
7	Medium	Fig. B.2(c)
8	Medium	Fig. B.2(d)
9	Medium	Fig. B.3(a)
10	Hard	Fig. B.3(b)
11	Hard	Fig. B.3(c)
12	Hard	Fig. B.3(d)
13	Hard	Fig. B.4(a)
14	Hard	Fig. B.4(b)

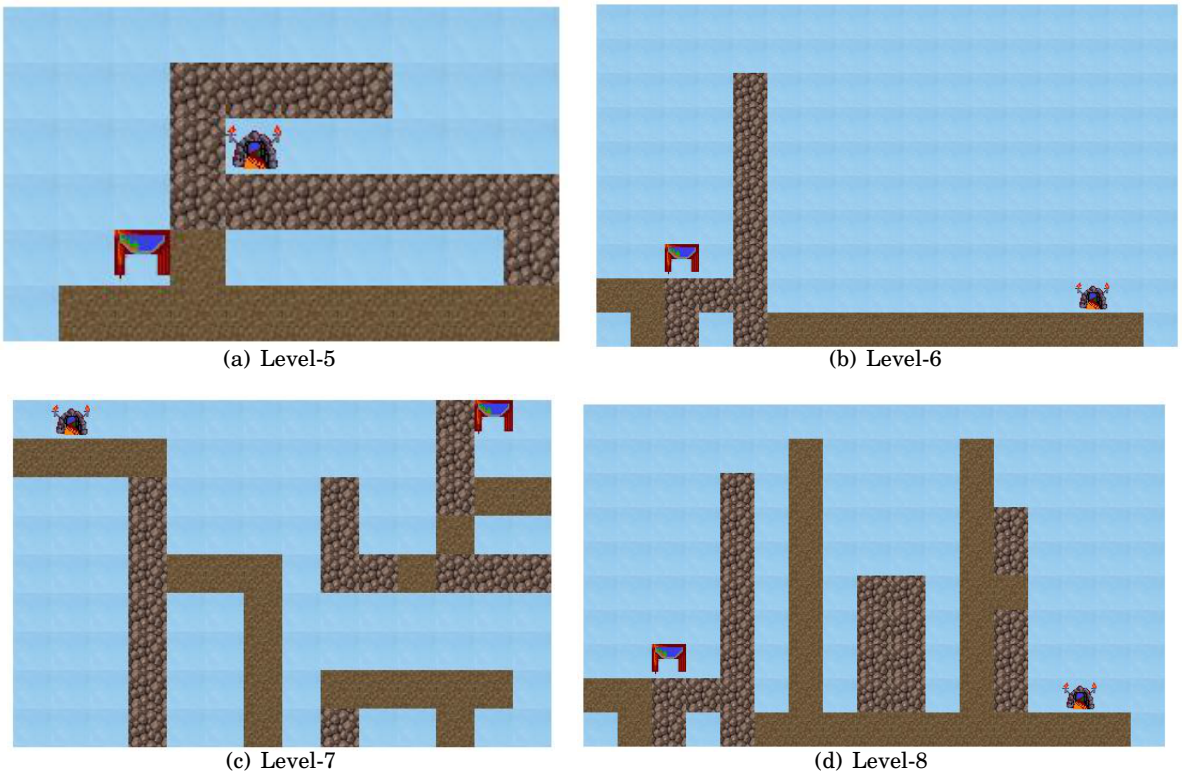
Table B.1: Complexity of each different level considered in this thesis.



(a) Level-1 (b) Level-2

(c) Level-3 (d) Level-4

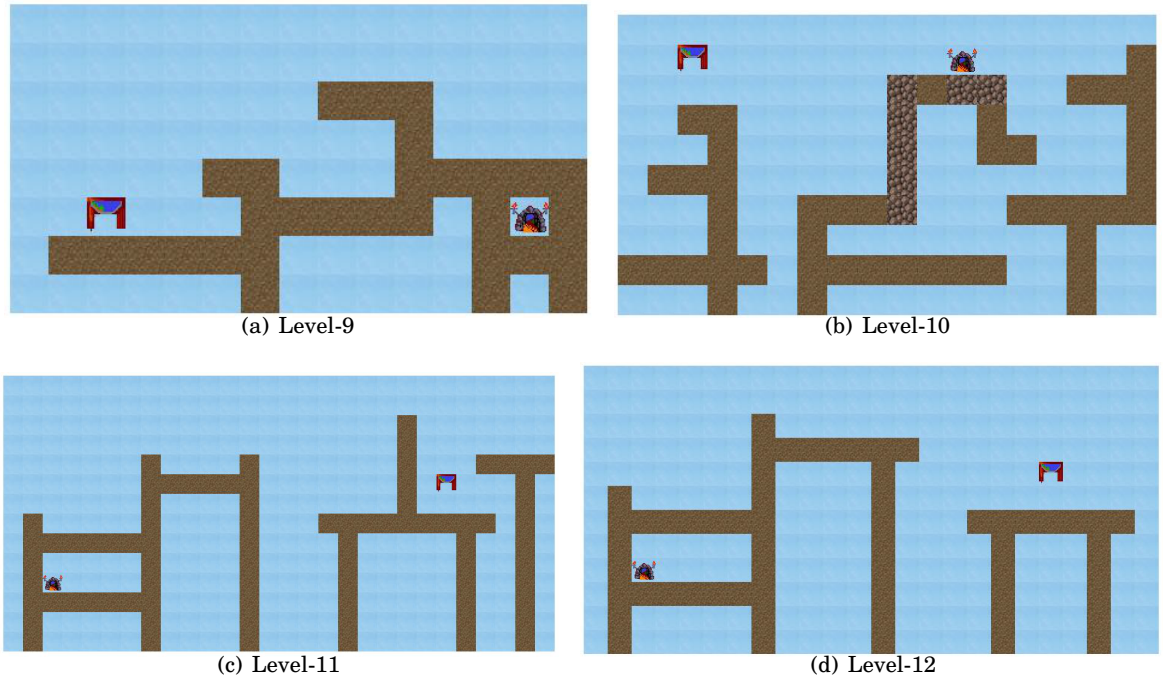
Figure B.1: Levels 1, 2, 3 and 4 designed for *The Lemmings* video game. All these levels are considered "Easy Levels".



(a) Level-5 (b) Level-6

(c) Level-7 (d) Level-8

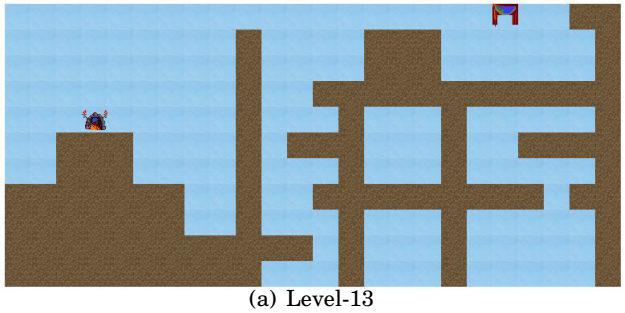
Figure B.2: This figure shows the levels 5, 6, 7 and 8.



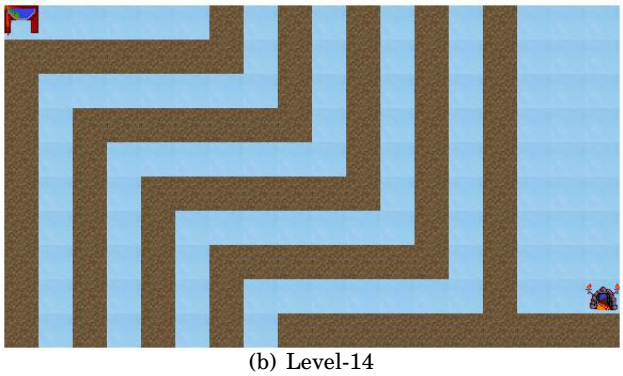
(a) Level-9 (b) Level-10

(c) Level-11 (d) Level-12

Figure B.3: The levels 9, 10, 11 and 12 are showed in this figure.



(a) Level-13



(b) Level-14

Figure B.4: This figure shows the hardest levels, 13 and 14, considered in this thesis.

Bibliography

[Gec, 2006] (2006). Gecode project. http://www.gecode.org/.

[cho, 2007] (2007). Choco web page. http://www.emn.fr/z-info/choco-solver/.

[MsP, 2007] (2007). Ms pac-man software kit. http://dces.essex.ac.uk/staff/sml/pacman/PacManContest.html.

[Moj, 2009] (2009). Infinite mario bros. available: https://mojang.com/notch/mario/.

[aiu, 2011] (2011). Aiur project. available: https://code.google.com/p/aiurproject/.

[201, 2011] (2011). An api for interacting with starcraft: Broodwar. available: https://code.google.com/p/bwapi/.

[Mar, 2012] (2012). Mario ai championship. available: http://www.marioai.org.

[bwa, 2012] (2012). A starcraft: Broodwar bot using bwapi. available: https://code.google.com/p/skynetbot/.

[bth, 2013] (2013). Bthai - a starcraft ai bot. available: https://code.google.com/p/bthai/.

[Abbattista et al., 1995] Abbattista, F., Abbattista, N., and Caponetti, L. (1995). An evolutionary and cooperative agents model for optimization. In *Evolutionary Computation, 1995., IEEE International Conference on*, volume 2, pages 668–671.

[Abraham and Ramos, 2003] Abraham, A. and Ramos, V. (2003). Web usage mining using artificial ant colony clustering and linear genetic programming. In *Evolutionary Computation, 2003. Congress on*, volume 2, pages 1384–1391 Vol.2.

[Agrawal et al., 1994] Agrawal, R. B., Deb, K., Deb, K., and Agrawal, R. B. (1994). Simulated binary crossover for continuous search space. Technical report.

[Alaya et al., 2004] Alaya, I., Solnon, C., and Ghedira, K. (2004). Ant algorithm for the multi-dimensional knapsack problem. In *International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2004*. Citeseer.

[Alhejali and Lucas, 2010] Alhejali, A. and Lucas, S. (2010). Evolving diverse ms. pac-man playing agents using genetic programming. In *Computational Intelligence (UKCI), 2010 UK Workshop on*, pages 1–6.

[Angeline, 1995] Angeline, P. J. (1995). Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence: A Dynamic Systems Perspective*, pages 152–163.

[Ates, 1989] Ates, R. (1989). Aggresive behaviour in corals. *Freshwater and Marine Aquarium*, 12(8):104–112.

[Bäck, 1994] Bäck, T. (1994). Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, volume 1, pages 57–62.

[Bäck et al., 1997] Bäck, T., Fogel, D. B., and Michalewicz, Z. (1997). *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK, 1st edition.

[Baker, 1987] Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, pages 14–21, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.

[Baran, 2010] Baran, B. (2010). Facebook as a formal instructional environment. *British Journal of Educational Technology*, 41(6).

[Barrero et al., 2010] Barrero, D. F., Gonzalez-Pardo, A., Camacho, D., and R-Moreno, M. D. (2010). Parameter tunning for genetic algorithm. *Computer Science and Information Systems (COMSIS)*, 7(3):661–677.

[Barrero et al., 2014] Barrero, D. F., Hernández-Castro, J. C., Peris-Lopez, P., and Camacho, D. (2014). A genetic tango attack against the david–prasad rfid ultra-lightweight authentication protocol. *Expert Systems*, 31(1):9–19.

[Bell and McMullen, 2004] Bell, J. E. and McMullen, P. R. (2004). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18(1):41–48.

[Bellotti et al., 2008] Bellotti, F., Berta, R., De Gloria, A., and Zappi, V. (2008). Exploring gaming mechanisms to enhance knowledge acquisition in virtual worlds. In *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts*, DIMEA '08, pages 77–84. ACM.

[Benedetto et al., 2002] Benedetto, D., Caglioti, E., and Loreto, V. (2002). Language Trees and Zipping. *Physical Review Letters*, 88(4):48702.

[Berns et al., 2013] Berns, A., Gonzalez-Pardo, A., and Camacho, D. (2013). Game-like language learning in 3-d virtual environments. *Computers and Education*, 60(1):210 – 220.

[Beyer and Schwefel, 2002] Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies - a comprehensive introduction. 1(1):3–52.

[Blazewicz et al., 1983] Blazewicz, J., Lenstra, J., and Kan, A. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11 – 24.

[Blickle and Thiele, 1996] Blickle, T. and Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evol. Comput.*, 4(4):361–394.

[Blum and Merkle, 2008] Blum, C. and Merkle, D. (2008). *Swarm Intelligence: Introduction and Applications*. Springer Publishing Company, Incorporated, 1 edition.

[Bonabeau et al., 1999] Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA.

[Boyd and Ellison, 2007] Boyd, D. and Ellison, N. B. (2007). Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230.

[Bremermann, 1962] Bremermann, H. J. (1962). Optimization through evolution and recombination. In *Proceedings of the Conference on Self-Organizing Systems – 1962*, pages 93–106, Washington, DC. Spartan Books.

[Bremermann et al., 1966] Bremermann, H. J., Rogson, M., and Salaff, S. (1966). Global properties of evolution processes. In *Natural Automata and Useful Simulations*, pages 3–41. Spartan Books.

[Bullnheimer et al., 1999] Bullnheimer, B., Hartl, R. F., and Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of operations research*, 89:319–328.

[Cadena and Garrido, 2011] Cadena, P. and Garrido, L. (2011). Fuzzy case-based reasoning for managing strategic and tactical reasoning in starcraft. In Batyrshin, I. Z. and Sidorov, G., editors, *MICAI (1)*, volume 7094 of *Lecture Notes in Computer Science*, pages 113–124. Springer.

[Cardona et al., 2013] Cardona, A., Togelius, J., and Nelson, M. (2013). Competitive coevolution in ms. pac-man. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1403–1410.

[Chadwick, 1987] Chadwick, N. E. (1987). Interspecific aggressive behavior of the corallimorpharian corynactis californica (cnidaria: Anthozoa): effects on sympatric corals and sea anemones. *The Biological Bulletin*, 173(1):110–125.

[Cilibrasi et al.,] Cilibrasi, R., Cruz, A. L., de Rooij, S., and Keijzer, M. *CompLearn Toolkit*. [Online] Available: http://www.complearn.org/.

[Cilibrasi and Vitanyi, 2005] Cilibrasi, R. and Vitanyi, P. (2005). Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545.

[Coldridge and Amos, 2010] Coldridge, J. and Amos, M. (2010). Genetic algorithms and the art of zen. *CoRR*, abs/1005.4446.

[Cormode, 2004] Cormode, G. (2004). The hardness of the lemmings game, or oh no, more np-completeness proofs. In *Proceedings of Third International Conference on Fun with Algorithms*, pages 65–76.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

[Costa and Hertz, 1997] Costa, D. and Hertz, A. (1997). Ants can colour graphs. *The Journal of the Operational Research Society*, 48(3):295–305.

[Darwin, 1859] Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection*. Murray, London.

[De Jong, 1975] De Jong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis.

[Deneubourg et al., 1990] Deneubourg, J.-L., Aron, S., and Goss, S. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3:159–169.

[Deng and Lin, 2011] Deng, G.-F. and Lin, W.-T. (2011). Ant colony optimization-based algorithm for airline crew scheduling problem. *Expert Syst. Appl.*, 38(5):5787–5793.

[Dorigo, 1992] Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms (in Italian)*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

[Dorigo and Gambardella, 1996] Dorigo, M. and Gambardella, L. (1996). A study of some properties of ant-q. In *Parallel Problem Solving from Nature — PPSN IV*, volume 1141 of *Lecture Notes in Computer Science*, pages 656–665. Springer Berlin Heidelberg.

[Dorigo and Gambardella, 1997a] Dorigo, M. and Gambardella, L. M. (1997a). Ant colonies for the traveling salesman problem. In *BioSystems*.

[Dorigo and Gambardella, 1997b] Dorigo, M. and Gambardella, L. M. (1997b). Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66.

[Dorigo et al., 1996] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *Systems Man and Cybernetics, Part B: Cybernetics. IEEE Transactions on*, 26(1):29–41.

[Dorigo and Stützle, 2001] Dorigo, M. and Stützle, T. (2001). An experimental study of the simple ant colony optimization algorithm. In *Advances in Fuzzy Systems and Evolutionary Computation*, Artificial Intelligence Series, pages 253–258. World Scientific and Engineering Society Press.

[Dowsland and Thompson, 2008] Dowsland, K. A. and Thompson, J. M. (2008). An improved ant colony optimisation heuristic for graph colouring. *Discrete Applied Mathematics*, 156(3):313 – 324.

[Eiben et al., 1994a] Eiben, A., Raué, P.-E., and Ruttkay, Z. (1994a). Genetic algorithms with multi-parent recombination. In *Parallel Problem Solving from Nature — PPSN III*, volume 866 of *Lecture Notes in Computer Science*, pages 78–87. Springer Berlin Heidelberg.

[Eiben et al., 1994b] Eiben, A., Raué, P.-E., and Ruttkay, Z. (1994b). Solving constraint satisfaction problems using genetic algorithms. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 542–547. IEEE.

[Eiben et al., 1999] Eiben, A. E., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 3(2):124–141.

[Eiben and Smith, 2009] Eiben, A. E. and Smith, J. E. (2009). *Introduction to Evolutionary Computing*. Springer-Verlag.

[Emilio et al., 2010] Emilio, M., Moises, M., Gustavo, R., and Yago, S. (2010). Pac-mant: Optimization based on ant colonies applied to developing an agent for ms. pac-man. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 458–464.

[Engelbrecht, 2007] Engelbrecht, A. P. (2007). *Computational Intelligence: An Introduction*. Wiley Publishing, 2nd edition.

[Eshelman et al., 1989] Eshelman, L. J., Caruana, R. A., and Schaffer, J. D. (1989). Biases in the crossover landscape. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 10–19, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Fernandes et al., 2008] Fernandes, C., Mora, A., Merelo, J. J., Ramos, V., Laredo, J., and Rosa, A. (2008). Kants: Artifical ant system for classification. In *Ant Colony Optimization and Swarm Intelligence*, volume 5217 of *Lecture Notes in Computer Science*, pages 339–346. Springer Berlin Heidelberg.

[Flensbank and Yannakakis, 2008] Flensbank, J. and Yannakakis, G. (2008). C# kit. available: http://mspacmanai.codeplex.com/.

[Fogel and Fogel, 1996] Fogel, D. and Fogel, L. (1996). An introduction to evolutionary programming. In *Artificial Evolution*, volume 1063 of *Lecture Notes in Computer Science*, pages 21–33. Springer Berlin Heidelberg.

[Fogel, 1995] Fogel, D. B. (1995). *Evolutionary computation: toward a new philosophy of machine intelligence*. IEEE Press.

[Fogel, 1962] Fogel, L. J. (1962). Autonomous automata. *Industrial Research*, 4:14–19.

[Fonseca and Fleming, 1998] Fonseca, C. M. and Fleming, P. J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 28(1):26–37.

[Forrest et al., 1994] Forrest, S., Perelson, A. S., Allen, L., and Cherukuri, R. (1994). Self-nonself discrimination in a computer. In *Proceedings of the IEEE Symposium in Security and Privacy*, SP '94, pages 202–212. IEEE Computer Society.

[França et al., 2008] França, F., Coelho, G. P., and Zuben, F. J. (2008). bicaco: An ant colony inspired biclustering algorithm. In *Ant Colony Optimization and Swarm Intelligence*, volume 5217 of *Lecture Notes in Computer Science*, pages 401–402. Springer Berlin Heidelberg.

[Fraser, 1957a] Fraser, A. S. (1957a). Simulation of genetic systems by automatic digital computers i: Effects of linkage on rates of advance under selection. *Australian Journal of Biological Sciences*, 10:492–499.

[Fraser, 1957b] Fraser, A. S. (1957b). Simulation of genetic systems by automatic digital computers i: Introduction. *Australian Journal of Biological Sciences*, 10:484–491.

[Freitas, 2003] Freitas, A. A. (2003). A survey of evolutionary algorithms for data mining and knowledge discovery. In *Advances in evolutionary computing*, pages 819–845. Springer Berlin Heidelberg.

[Gambardella and Dorigo, 1996] Gambardella, L. and Dorigo, M. (1996). Solving symmetric and asymmetric tsps by ant colonies. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 622–627.

[Gambardella et al., 1999] Gambardella, L., Taillard, E., and Dorigo, M. (1999). Ant colonies for the qap. *Journal of the Operational Research Society*, 50:167–176.

[Garrison and Kanuka, 2004] Garrison, D. and Kanuka, H. (2004). Blended learning: Uncovering its transformative potential in higher education. *The Internet and Higher Education*, 7(2):95–105.

[Genin et al., 1994] Genin, A., Karp, L., and Miroz, A. (1994). Effects of flow on competitive superiority in scleractinian corals. *Limnology and Oceanography*, 39(4):913–924.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1st edition.

[Goldberg and Deb, 1991] Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann.

[Goldberg and Lingle, 1985] Goldberg, D. E. and Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications*, pages 154–159. Lawrence Erlbaum Associates, Publishers.

[Gonçalves et al., 2008] Gonçalves, J. F., Mendes, J. J., and Resende, M. G. (2008). A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research*, 189(3):1171–1190.

[Gonzalez-Pardo et al., 2010a] Gonzalez-Pardo, A., Barrero, D. F., Camacho, D., and R-Moreno, M. D. (2010a). A case study on grammatical-based representation for regular expression evolution. In *Trends in Practical Applications of Agents and Multiagent Systems*, volume 71 of *Advances in Soft Computing*, pages 379–386. Springer Berlin / Heidelberg.

[Gonzalez-Pardo and Camacho, 2011] Gonzalez-Pardo, A. and Camacho, D. (2011). Analysis of grammatical evolution approaches to regular expression induction. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 639 – 646.

[Gonzalez-Pardo and Camacho, 2013a] Gonzalez-Pardo, A. and Camacho, D. (2013a). Environmental influence in bio-inspired game level solver algorithms. In *Proceedings of the 7th International Symposium on Intelligent Distributed Computing - IDC 2013*, Studies in Computational Intelligence, page In press. Springer Berlin Heidelberg.

[Gonzalez-Pardo and Camacho, 2013b] Gonzalez-Pardo, A. and Camacho, D. (2013b). Maximal component detection in graphs using swarm-based and genetic algorithms. In Fortino, G., Badica, C., Malgeri, M., and Unland, R., editors, *Intelligent Distributed Computing VI*, volume 446 of *Studies in Computational Intelligence*, pages 247–252. Springer Berlin Heidelberg.

[Gonzalez-Pardo and Camacho, 2013c] Gonzalez-Pardo, A. and Camacho, D. (2013c). A new csp graph-based representation for ant colony optimization. In *2013 IEEE Conference on Evolutionary Computation*, volume 1, pages 689–696.

[Gonzalez-Pardo and Camacho, 2014a] Gonzalez-Pardo, A. and Camacho, D. (2014a). Designing an aco model for clustering student behaviours in virtual worlds. *Integrated Computer-Aided Engineering (ICAE).*, Submitted.

[Gonzalez-Pardo and Camacho, 2014b] Gonzalez-Pardo, A. and Camacho, D. (2014b). A new csp graph-based representation to resource-constrained project scheduling problem. In *2014 IEEE Conference on Evolutionary Computation*, page In press.

[Gonzalez-Pardo et al., 2010b] Gonzalez-Pardo, A., de Borja Rodriguez, F., Pulido, E., and Camacho, D. (2010b). Using virtual worlds for behaviour clustering-based analysis. In *ACM Workshop on Surreal Media and Virtual Cloning*, pages 9 – 14. ACM.

[Gonzalez-Pardo et al., 2010c] Gonzalez-Pardo, A., Granados, A., Camacho, D., and de Borja Rodriguez, F. (2010c). Influence of music representation on compression-based clustering. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2988 – 2995. IEEE Xplore.

[Gonzalez-Pardo et al., 2014a] Gonzalez-Pardo, A., Palero, F., and Camacho, D. (2014a). An empirical study on collective intelligence algorithms for video games problem-solving. *Computing and Informatics*, In press.

[Gonzalez-Pardo et al., 2014b] Gonzalez-Pardo, A., Palero, F., and Camacho, D. (2014b). Micro and macro lemmings simulations based on ants colonies. In *EvoApplications 2014*, page In press.

[Gonzalez-Pardo et al., 2014c] Gonzalez-Pardo, A., Rosa, A., and Camacho, D. (2014c). Behaviour-based identification of student communities in virtual worlds. *Computer Science and Information Systems*, 11(1):195–213.

[Gonzalez-Pardo et al., 2012] Gonzalez-Pardo, A., Varona, P., Camacho, D., and de Borja Rodriguez, F. (2012). Communication by identity in bio-inspired multi-agent systems. *International Journal Concurrency and Computation: Practice & Experience*, 2012(24):589–603.

[Goss et al., 1989] Goss, S., Aron, S., Deneubourg, J., and Pasteels, J. (1989). Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12):579–581.

[Gottlieb et al., 2003] Gottlieb, J., Puchta, M., and Solnon, C. (2003). A study of greedy, local search, and ant colony optimization approaches for car sequencing problems. In *Applications of Evolutionary Computing*, volume 2611 of *Lecture Notes in Computer Science*, pages 246–257. Springer Berlin Heidelberg.

[Grassé, 1959] Grassé, P.-P. (1959). La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6(1):41–80.

[Guntsch and Middendorf, 2001] Guntsch, M. and Middendorf, M. (2001). Pheromone modification strategies for ant algorithms applied to dynamic tsp. In Boers, E., editor, *Applications of Evolutionary Computing*, volume 2037 of *Lecture Notes in Computer Science*, pages 213–222. Springer Berlin Heidelberg.

[Halkidi et al., 2002] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2002). Cluster validity methods: Part i. *SIGMOD Record*, 31(2):40–45.

[Hamdi et al., 2008] Hamdi, A., Monmarché, N., Alimi, M., and Slimane, M. (2008). Swarmclass: A novel data clustering approach by a hybridization of an ant colony with flying insects. In *Ant Colony Optimization and Swarm Intelligence*, volume 5217 of *Lecture Notes in Computer Science*, pages 411–412. Springer Berlin Heidelberg.

[Handl and Meyer, 2007] Handl, J. and Meyer, B. (2007). Ant-based and swarm-based clustering. *Swarm Intelligence*, 1(2):95–113.

[Hardy, 1996] Hardy, A. (1996). On the number of clusters. *Computational Statistics & Data Analysis*, 23(1):83–96.

[Hartmann, 1998] Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7):733–750.

[Helmer, 2007] Helmer, S. (2007). Measuring the structural similarity of semistructured documents using entropy. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 1022–1032. VLDB Endowment.

[Hinterding, 1995] Hinterding, R. (1995). Gaussian mutation and self-adaption for numeric genetic algorithms. In *Evolutionary Computation, 1995., IEEE International Conference on*, volume 1, pages 384–.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

[Holland, 1992] Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366.

[Hoseini Semnani and Zamanifar, 2012] Hoseini Semnani, S. and Zamanifar, K. (2012). The power of ants in solving distributed constraint satisfaction problems. *Appl. Soft Comput.*, 12(2):640–651.

[Hou et al., 2011] Hou, N. C., Hong, N. S., On, C. K., and Teo, J. (2011). Infinite mario bross ai using genetic algorithm. In *Sustainable Utilization and Development in Engineering and Technology (STUDENT), 2011 IEEE Conference on*, pages 85–89.

[Hruschka et al., 2009] Hruschka, E. R., Campello, R. J. G. B., Freitas, A. A., and De Carvalho, A. C. P. L. F. (2009). A survey of evolutionary algorithms for clustering. *Trans. Sys. Man Cyber Part C*, 39(2):133–155.

[Ishioka, 2000] Ishioka, T. (2000). Extended k-means with an efficient estimation of the number of clusters. In *Intelligent Data Engineering and Automated Learning - IDEAL 2000, Data Mining, Financial Engineering, and Intelligent Agents, Second International Conference, Shatin, N.T. Hong Kong, China, December 13-15, 2000, Proceedings*, pages 17–22. Springer.

[Jabbarpour et al., 2014] Jabbarpour, M. R., Malakooti, H., Noor, R. M., Anuar, N. B., and Khamis, N. (2014). Ant colony optimisation for vehicle traffic systems: applications and challenges. *Int. J. of Bio-Inspired Computation*, 6(1):32 – 56.

[Kanungo et al., 2002] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892.

[Karaboga and Basturk, 2007] Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471.

[Ke et al., 2010] Ke, L., Feng, Z., Ren, Z., and Wei, X. (2010). An ant colony optimization approach for the multidimensional knapsack problem. *Journal of Heuristics*, 16(1):65–83.

[Kendall and Spoerer, 2004] Kendall, G. and Spoerer, K. (2004). Scripting the game of lemmings with a genetic algorithm. In *Proc of the 2004 IEEE Congress on Evolutionary Computation*, pages 117–124.

[Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948.

[Khan et al., 2009] Khan, S., Bilal, M., Sharif, M., Sajid, M., and Baig, R. (2009). Solution of n-queen problem using aco. In *IEEE 13th International Multitopic Conference, INMIC. 2009*.

[Kolisch and Hartmann, 2006] Kolisch, R. and Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37.

[Kolisch and Sprecher, 1996] Kolisch, R. and Sprecher, A. (1996). Psplib – a project scheduling problem library. *European Journal of Operational Research*, 96:205–216.

[Kosko, 1992] Kosko, B. (1992). *Neural networks and fuzzy systems*. Prentice hall.

[Koza, 1992] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

[Kraskov et al., 2005] Kraskov, A., Stoegbauer, H., Andrzejak, R., and Grassberger, P. (2005). Hierarchical clustering using mutual information. *Europhysics Letters*, 70(2):278–284.

[Lai and Liaw, 2008] Lai, J. Z. C. and Liaw, Y.-C. (2008). Improvement of the k-means clustering filtering algorithm. *Pattern Recogn.*, 41(12):3677–3681.

[Lara-Cabrera et al., 2014] Lara-Cabrera, R., Cotta, C., and Fernández-Leiva, A. J. (2014). On balance and dynamism in procedural content generation with self-adaptive evolutionary algorithms. *Natural Computing*, pages 1–12.

[Liberatore et al., 2014] Liberatore, F., Mora, A., Castillo, P., and Merelo, J. J. (2014). Evolving evil: Optimizing flocking strategies through genetic algorithms for the ghost team in the game of ms. pac-man. In *EvoApplications 2014*, page In press.

[Lipka, 2007] Lipka, S. (2007). For professors, 'friending' can be fraught. (cover story). *Chronicle of Higher Education*, 54(15).

[Lucas, 2009] Lucas, S. M. (2009). Computational intelligence and ai in games: A new ieee transactions. *IEEE Trans. Comput. Intellig. and AI in Games*, 1(1):1–3.

[MacKay, 2003] MacKay, D. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

[Macqueen, 1967] Macqueen, J. B. (1967). Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

[Madge et al., 2009] Madge, C., Meek, J., Wellens, J., and Hooley, T. (2009). Facebook, social integration and informal learning at university: "it is more for socialising and talking to friends about work than for actually doing work". *Learning Media And Technology*, 34(2):141–155.

[Maniezzo and Colorni, 1999] Maniezzo, V. and Colorni, A. (1999). The ant system applied to the quadratic assignment problem. *Knowledge and Data Engineering, IEEE Transactions on*, 11(5):769–778.

[Marais, 1937] Marais, E. N. (1937). *Die Siel van die Mier (The Soul of the Ant)*.

[Martin et al., 2010] Martin, E., Martinez, M., Recio, G., and Saez, Y. (2010). Pac-mant: Optimization based on ant colonies applied to developing an agent for ms. pac-man. In Yannakakis, G. N. and Togelius, J., editors, *CIG*, pages 458–464. IEEE.

[Martın-Bautista et al., 1999] Martın-Bautista, M. J., Vila, M. A., and Larsen, H. L. (1999). A fuzzy genetic algorithm approach to an adaptive information retrieval agent. *Journal of the American Society for Information Science*, 50(9):760–771.

[Maulik and Bandyopadhyay, 2000] Maulik, U. and Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455 – 1465.

[McPartland and Gallagher, 2012] McPartland, M. and Gallagher, M. (2012). Interactively training first person shooter bots. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 132–138.

[Merelo et al., 2013a] Merelo, J. J., Mora, A., Cotta, C., and Fernández-Leiva, A. J. (2013a). Finding an evolutionary solution to the game of mastermind with good scaling behavior. In *Learning and Intelligent Optimization*, pages 288–293. Springer Berlin Heidelberg.

[Merelo et al., 2013b] Merelo, J. J., Mora, A., Cotta, C., and Rico, N. (2013b). Influence of selective pressure on quality of solutions and speed of evolutionary mastermind. In *Foundations of Computational Intelligence (FOCI), 2013 IEEE Symposium on*, pages 122–129. IEEE.

[Merkle et al., 2002] Merkle, D., Middendorf, M., and Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *Evolutionary Computation, IEEE Transactions on*, 6(4):333–346.

[Michalewicz, 1996] Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, London, UK, UK.

[Min and Siqing, 2010] Min, W. and Siqing, Y. (2010). Improved k-means clustering based on genetic algorithm. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, volume 6, pages V6–636 –V6–639.

[Mora et al., 2014] Mora, A., Merelo, J. J., García-Sánchez, P., Castillo, P., Rodríguez-Domingo, M. S., and Hidalgo-Bermúdez, R. M. (2014). Creating autonomous agents for playing super mario bros game by means of evolutionary finite state machines. *Evolutionary Intelligence*, 6(4):205–218.

[Morin et al., 2009] Morin, S., Gagné, C., and Gravel, M. (2009). Ant colony optimization with a specialized pheromone trail for the car-sequencing problem. *European Journal of Operational Research*, 197:1185 – 1191.

[Nemes and Roska, 1995] Nemes, L. and Roska, T. (1995). A cnn model of oscillation and chaos in ant colonies: A case study. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 42(10):741–745.

[Noor Shaker and Yannakakis, 2012a] Noor Shaker, J. T. and Yannakakis, G. (2012a). Mario ai. level generation track. available: http://www.marioai.org/LevelGeneration.

[Noor Shaker and Yannakakis, 2012b] Noor Shaker, J. T. and Yannakakis, G. (2012b). Mario ai. turing test track. available: http://www.marioai.org/turing-test-track.

[Ontañón et al., 2013] Ontañón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., and Preuss, M. (2013). A survey of real-time strategy game ai research and competition in starcraft. *IEEE Trans. Comput. Intellig. and AI in Games*, 5(4):293–311.

[Ozdamar, 1999] Ozdamar, L. (1999). A genetic algorithm approach to a general category project scheduling problem. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 29(1):44–59.

[Palero et al., 2014] Palero, F., Gonzalez-Pardo, A., and Camacho, D. (2014). Simple gamer interaction analysis through tower defence games. In *6th International Conference on Computational Collective Intelligence Technologies and Applications (ICCCI 2014)*, page In press.

[Parpinelli et al., 2002] Parpinelli, R. S., Lopes, H. S., and Freitas, A. A. (2002). Data mining with an ant colony optimization algorithm. *Evolutionary Computation, IEEE Transactions on*, 6(4):321–332.

[Pavlov, 1998] Pavlov, I. (1998). *LZMAX*. [Online] Available: http://www.7-zip.org/sdk.html.

[Pelleg and Moore, 2000] Pelleg, D. and Moore, A. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *In Proceedings of the 17th International Conf. on Machine Learning*, pages 727–734. Morgan Kaufmann.

[Perez, 2013] Perez, D. (2013). Ptsp framework. available: http://www.ptsp-game.net/code.php?c=league.

[Perez et al., 2014] Perez, D., Powley, E. J., Whitehouse, D., Rohlfshagen, P., Samothrakis, S., Cowling, P. I., and Lucas, S. M. (2014). Solving the physical traveling salesman problem: Tree search and macro actions. *IEEE Trans. Comput. Intellig. and AI in Games*, 6(1):31–45.

[Pham et al., 2005] Pham, D. T., Dimov, S. S., and Nguyen, C. D. (2005). Selection of k in k -means clustering. *Proceedings of the I MECH E Part C Journal of Mechanical Engineering Science*, 219(1):103–119.

[Pimont and Solnon, 2000] Pimont, S. and Solnon, C. (2000). A generic ant algorithm for solving constraint satisfaction problems. *Abstract proceedings of ANTS*, pages 100–108.

[Poli et al., 2007] Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57.

[Poli et al., 2008] Poli, R., Langdon, W. B., and McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd.

[Prabha and Saranya, 2011] Prabha, K. and Saranya, R. (2011). Refinement of k-means clustering using genetic algorithm. *Journal of Computer Applications (JCA)*, 4(2):40 – 44.

[Preuss et al., 2013] Preuss, M., Garcia, A. M., and Mahlmann, T. (2013). Starcraft rts ai competition. available: http://ls11-www.cs.uni-dortmund.de/rts-competition/starcraft-cig2013.

[Prins, 2004] Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985 – 2002.

[Ramirez-Atencia et al., 2014] Ramirez-Atencia, C., Bello-Orgaz, G., R-Moreno, M., and Camacho, D. (2014). A simple csp-based model for unmanned air vechicle mission planning. In *IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA 2014)*, page In press.

[Rechenberg, 1965] Rechenberg, I. (1965). *Cybernetic solution path of an experimental problem*. Royal Aircraft Establishment, Farnborough.

[Reed et al., 1967] Reed, J., Toombs, R., and Barricelli, N. A. (1967). Simulation of biological evolution and machine learning. *Journal of Theoretical Biology*, 17:319–342.

[Reisinger et al., 2007] Reisinger, J., Bahçeci, E., Karpov, I., and Miikkulainen, R. (2007). Coevolving strategies for general game playing. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games, CIG 2007, Honolulu, Hawaii, USA, 1-5 April, 2007*, pages 320–327. IEEE.

[Ritzema and Harris, 2008] Ritzema, T. and Harris, B. (2008). The use of second life for distance education. *J. Comput. Sci. Coll.*, 23(6):110–116.

[Rizzoli et al., 2007] Rizzoli, A., Montemanni, R., Lucibello, E., and Gambardella, L. (2007). Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, 1(2):135–151.

[Roli et al., 2001] Roli, A., Blum, C., and Dorigo, M. (2001). ACO for maximal constraint satisfaction problems. In *Proceedings of MIC'2001 – Metaheuristics International Conference*, volume 1, pages 187–191, Porto, Portugal.

[Salcedo-Sanz et al., 2013a] Salcedo-Sanz, S., del Ser, J., Landa-Torres, I., Gil-López, S., and Portilla-Figueras, J. A. (2013a). The coral reefs optimization algorithm: an efficient meta-heuristic for solving hard optimization problems. In *15th Applied Stochastic Models and Data Analysis (ASMDA 2013)*, pages 751 – 758.

[Salcedo-Sanz et al., 2013b] Salcedo-Sanz, S., Pastor-Sánchez, A., Gallo-Marazuela, D., and Portilla-Figueras, A. (2013b). A novel coral reefs optimization algorithm for multi-objective problems. In *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, volume 8206 of *Lecture Notes in Computer Science*, pages 326–333. Springer Berlin Heidelberg.

[Salcedo-Sanz et al., 2013c] Salcedo-Sanz, S., Sanchez-Garcia, J. E., Portilla-Figueras, J. A., Jimenez-Fernandez, S., and Ahmadzadeh, A. M. (2013c). A coral-reef optimization algorithm for the optimal service distribution problem in mobile radio access networks. *Transactions on Emerging Telecommunications Technologies*, pages n.a. – n.a.

[Schaffer and Morishima, 1987] Schaffer, J. D. and Morishima, A. (1987). An adaptive crossover distribution mechanism for genetic algorithms. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, pages 36–40.

[Selwyn and Grant, 2009] Selwyn, N. and Grant, L. (2009). Researching the realities of social software use – an introduction. *Learning, Media and Technology*, 34:79 – 86.

[Sinrod, 2007] Sinrod, E. J. (2007). Virtual world litigation for real.

[Solnon, 2002] Solnon, C. (2002). Ants can solve constraint satisfaction problems. *IEEE Transactions on Evolutionary Computation*, 6:347–357.

[Solnon, 2010] Solnon, C. (2010). *Ant Colony Optimization and Constraint Programming*. Wiley-ISTE.

[Solnon et al., 2008] Solnon, C., Cung, V. D., Nguyen, A., and Artigues, C. (2008). The car sequencing problem: overview of state-of-the-art methods and industrial case-study of the roadefâ€™2005 challenge problem. *European Journal of Operational Research*, 191(3):912–927.

[Sorlin and Solnon, 2004] Sorlin, S. and Solnon, C. (2004). A global constraint for graph isomorphism problems. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 287–301. Springer Berlin Heidelberg.

[Srinivas and Patnaik, 1994] Srinivas, M. and Patnaik, L. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(4):656–667.

[Stützle, 1998] Stützle, T. (1998). Parallelization strategies for ant colony optimization. In Eiben, A., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature — PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 722–731. Springer Berlin Heidelberg.

[Suomalainen, 2012] Suomalainen, P. (2012). Clustering moodle data via ant colony optimization. In *Proceedings of the 8th International Conference on Swarm Intelligence*, ANTS'12, pages 340–341, Berlin, Heidelberg. Springer-Verlag.

[Talbot, 2007] Talbot, D. (2007). The fleecing of the avatars.

[Telles et al., 2007] Telles, G., Minghim, R., and Paulovich, F. (2007). Normalized compression distance for visual analysis of document collections. *Computers & Graphics*, 31(3):327–337.

[Teodorović and Dell'Orco, 2005] Teodorović, D. and Dell'Orco, M. (2005). Bee colony optimization–a cooperative learning approach to complex transportation problems. In *Advanced OR and AI Methods in Transportation: Proceedings of 16th Mini–EURO Conference and 10th Meeting of EWGT (13-16 September 2005).–Poznan: Publishing House of the Polish Operational and System Research*, pages 51–60.

[Togelius et al., 2009] Togelius, J., Karakovskiy, S., Koutnik, J., and Schmidhuber, J. (2009). Super mario evolution. pages 156–161.

[Togelius et al., 2013] Togelius, J., Shaker, N., Karakovskiy, S., and Yannakakis, G. N. (2013). The mario ai championship 2009-2012. *AI Magazine*, 34(3):89–92.

[Toth and Vigo, 2001] Toth, P. and Vigo, D. (2001). *The vehicle routing problem*. Siam.

[Tsang, 1993] Tsang, E. P. K. (1993). *Foundations of constraint satisfaction.* Computation in cognitive science. Academic Press.

[Turing, 1936] Turing, A. (1936). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265.

[Usluel and Mazman, 2009] Usluel, Y. K. and Mazman, S. G. (2009). Adoption of web 2.0 tools in distance education. *Procedia Social and Behavioral Sciences*, 1(1):818–823.

[Wang, 2014] Wang, Y. (2014). The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem. *Computers & Industrial Engineering*, 70:124–133.

[Wu and Yu, 2009] Wu, J. and Yu, W. (2009). Optimization and improvement based on k-means cluster algorithm. In *Knowledge Acquisition and Modeling, 2009. KAM '09. Second International Symposium on*, volume 3, pages 335 –339.

[Xing et al., 2010] Xing, L., Chen, Y., Wang, P., Zhao, Q., and Xiong, J. (2010). A knowledge-based ant colony optimization for flexible job shop scheduling problems. *Applied Soft Computing*, 10(3):888–896.

[Young and Hawes, 2012] Young, J. and Hawes, N. (2012). Evolutionary learning of goal priorities in a real-time strategy game. In Riedl, M. and Sukthankar, G., editors, *AIIDE*. The AAAI Press.

[Zampelli et al., 2010] Zampelli, S., Deville, Y., and Solnon, C. (2010). Solving subgraph isomorphism problems with constraint programming. *Constraints*, 15(3):327–353.

[Zhang et al., 2007] Zhang, X., Hao, Y., Zhu, X., and Li, M. (2007). Information distance from a question to an answer. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 874–883, New York, NY, USA. ACM.